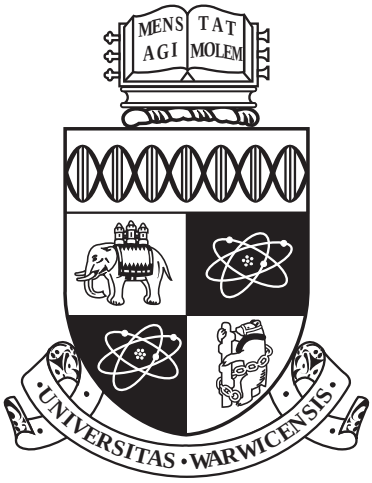

New Algorithms for Distributed Submodular Maximization

by

Rafael da Ponte Barbosa

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Department of Computer Science

February 2017

THE UNIVERSITY OF
WARWICK

Contents

Acknowledgments

I am grateful to my supervisor, Alina Ene, for all the guidance provided during my doctoral studies. Alina was (and is still) a true inspiration, and always had the right advice to give.

I thank the Department of Computer Science and the Centre for Discrete Mathematics and its Applications (DIMAP), at the University of Warwick, for all the opportunities presented during my studies. In particular, I would like to thank Artur Czumaj and Dan Král for all the help and support throughout these years.

I want to thank all the amazing people I have met at Warwick for making my time more enjoyable. To name a few, in no particular order, Jan Volec, Andrzej Grzesik, Lukáš Mach, Michail Fasoulakis, Taísa Martins, Nicholas Matsakis, Matthew Felice-Pace, Lehilton Pedrosa and Justin Ward.

Last but not least, I would like to thank my family for their unconditional love and support. I would not have made it here without them.

Declarations

The core of this thesis is based on new results, most of which have already been published. Below, we list those.

- Chapter 4 is mostly based on [da Ponte Barbosa et al., 2015], presented at the 32$^{\text{nd}}$ International Conference on Machine Learning (ICML), in 2015.
- Chapter 5 is mostly based on [da Ponte Barbosa et al., 2016], presented at the 57$^{\text{th}}$ Annual IEEE Symposium on Foundations of Computer Science (FOCS), in 2016.
- Chapter 6 is based on unpublished work.

None of the work presented in this thesis has been submitted for a previous degree at any university. All work is based on collaboration with Alina Ene, Huy L. Nguyen and Justin Ward, and was conducted during my period of study in the University of Warwick.

Abstract

A wide variety of problems in machine learning, including exemplar clustering, document summarization, and sensor placement, can be cast as submodular maximization problems. In many of these applications, the amount of data collected is quite large and it is growing at a very fast pace. For example, the wide deployment of sensors has led to the collection of large amounts of measurements of the physical world. Similarly, medical data and human activity data are being captured and stored at an ever increasing rate and level of detail. This data is often high-dimensional and complex, and it needs to be stored and/or processed in a distributed fashion. Following a recent line of work, we present here parallel algorithms for these problems, and analyze the compromise between quality of the solutions obtained and the amount of computational overhead.

On the one hand, we develop strategies for bringing existing algorithms for constrained submodular maximization in the sequential setting to the distributed setting. The algorithms presented achieve constant approximation factors in two rounds, and near optimal approximation ratios in only a constant number of rounds. Our techniques also give a fast sequential algorithm for non-monotone maximization subject to a matroid constraint.

On the other hand, for unconstrained submodular maximization, we devise parallel algorithms combining naive random sampling and Double Greedy steps, and investigate how much the quality of the solutions degrades with less coordination.

Chapter 1

Introduction

This thesis considers distributed algorithms for a class of discrete optimization problems. The problems investigated here have the following general structure. Given a submodular function f, and (possibly) a set of constraints, we aim to find a feasible set S maximizing $f(S)$. As demonstrated by the extensive prior works on submodular maximization, the community has a good understanding of the problem under remarkably general settings, which are handled by a small collection of general (sequential) algorithms. Nonetheless, many of today's applications of these problems involve processing enormous datasets, which require efficient, distributed algorithms.

Submodularity is a central concept in combinatorial optimization. Suppose we have a collection of elements V, and a function f that assigns a value $f(S)$ to every subset S of V. The function f is said to be *submodular* if it captures the *diminishing returns property*: the marginal gain of adding an element to a set is non-increasing as the set grows larger. This concept comprise a broad class of functions that arise in both theoretical and practical contexts. From a theoretical perspective, the class of submodular functions is extremely rich, including examples as varied as cut functions of graphs and digraphs, the Shannon entropy function, weighted coverage functions, and log-determinants. From an applied viewpoint, there has been a great deal of interest in practical applications of submodular optimization. Variants of facility location, sampling, sensor selection, clustering, influence maximization in social networks, and welfare maximization problems are all instances of submodular maximization.

The problem of maximizing a submodular function has been shown to be NP-hard for many of its classes, such as maximum weighted coverage or maximum cut in a graph. This motivated the study of approximation algorithms for the problem, starting with the standard greedy algorithm, already in the seventies [Nemhauser et al., 1978a,b]. It follows a simple strategy: starting with an empty set, while there is an element that

can be included in that solution (maintaining feasibility), pick the one with the largest marginal gain. Despite its simplicity, the greedy algorithm was shown to attain very good (sometimes tight) approximation guarantees. In light of these accomplishments, most successful approaches known today for submodular maximization problems have been based on the sequential greedy algorithm, including the continuous greedy algorithm [Calinescu et al., 2011; Feldman et al., 2011], and the double greedy algorithm [Buchbinder et al., 2012]. Such approaches attain the best-possible, tight approximation guarantees in a variety of settings [Feige, 1998; Vondrák, 2009; Dobzinski and Vondrák, 2012].

In many real-world applications, the amount of data that is collected nowadays is quite large and is growing at a very fast pace. For example, the wide deployment of sensors has led to the collection of large amounts of measurements of the physical world [Krause and Guestrin, 2007]. Similarly, medical data and human activity data are being captured and stored at an ever increasing rate and level of detail [Tsanas et al., 2010]. These large datasets require scalable approaches, such as distributed algorithms. Moreover, we look for approaches that reconcile competing goals: the solutions should be competitive compared to the centralized solution on the entire dataset, but the computations needed to find such solutions across several machines should use minimal amount of communication and synchronization.

In search of inspiration for distributed algorithms for submodular maximization problems, one may turn to the aforementioned successful strategies. However, they all share a common limitation, inherited from the standard greedy algorithm: they are inherently sequential. Indeed, the standard greedy algorithm repeatedly selects a feasible element that gives the largest marginal increase in objective value *with respect to the elements previously chosen*. Similarly, the continuous greedy algorithm chooses a direction in each discrete time step based on the gain with respect to some current fractional solution, and the double greedy algorithm makes a randomized choice for each element based on probabilities that depend on the choices made for all previous elements. This presents a seemingly fundamental barrier to obtaining efficient, highly parallel variants of these algorithms.

In this thesis, we make progress in addressing this question, describing parallel algorithms for submodular maximization that obtain provable approximation guarantees. Specifically, we study ways to employ the sequential (greedy) approaches as subroutines in a distributed algorithm. We analyze the trade-off between the quality of the solution

obtained and the communication overhead.

1.1 Thesis organization

Chapters 2 and 3 are introductory. The first presents definitions and notation used in the text; relevant related work is also included in Section 2.5. The second gives an overview of well-known algorithms used in the remainder of the thesis, as well as some important analyses.

Chapters 4 and 5 are concerned with constrained submodular maximization in a distributed setting. The results obtained in those are summarized in Table 1.1.

In the former chapter, building on the work of Mirzasoleiman et al. [2013], we develop a simple distributed (randomized) algorithm that runs in two MapReduce rounds and achieves provable, constant factor approximation guarantees. The techniques used also give fast sequential algorithms.

In the latter, we further extend those ideas and develop a generic framework for utilizing sequential algorithms in a distributed setting. Roughly speaking, our framework almost recovers the approximation guarantees of the sequential in a constant number of rounds.

Chapter 6 deals with results for the unconstrained case in the parallel transaction processing systems model. Inspired by [Pan et al., 2014], we develop parallel algorithms mixing random sampling and double greedy steps, and study the trade-off between level of parallelization and approximation guarantee obtained. Given a parameter p, which represents the portion of the elements to be processed serially through double greedy steps, our algorithms attain approximation guarantees of $\max\left\{\frac{1}{4}+\frac{p}{8},\ \frac{p}{2}\right\}$ and $\frac{1}{4}+\frac{p}{8}+\frac{p^2}{8}$.

Finally, we include in Chapter 7 some concluding remarks. In the appendix, we show an improved analysis of the distributed algorithm of Mirzasoleiman et al. [2013].

Monotone functions			
Constraint	Rounds	Approx.	Citation
cardinality	$O\left(\frac{\log \Delta}{\epsilon}\right)$	$1 - \frac{1}{e} - \epsilon$	Kumar et al. [2013]
	2	0.545	Mirrokni and Zadimoghaddam [2015]
	$O\left(\frac{1}{\epsilon}\right)$	$1 - \frac{1}{e} - \epsilon$	Theorem 5.2.1
matroid	$O\left(\frac{\log \Delta}{\epsilon}\right)$	$\frac{1}{2} - \epsilon$	Kumar et al. [2013]
	2	$\frac{1}{4}$	Corollary 4.2.4
	$O\left(\frac{1}{\epsilon}\right)$	$1 - \frac{1}{e} - \epsilon$	Theorem 5.3.3
p-system	$O\left(\frac{\log \Delta}{\epsilon}\right)$	$\frac{1}{p+1} - \epsilon$	Kumar et al. [2013]
	2	$\frac{1}{2(p+1)}$	Corollary 4.2.4
	$O\left(\frac{1}{\epsilon}\right)$	$\frac{1}{p+1} - \epsilon$	Theorem 5.2.1

Non-monotone functions			
Constraint	Rounds	Approx.	Citation
cardinality	2	$\frac{1-\frac{1}{m}}{2+e}$	Mirrokni and Zadimoghaddam [2015]
	2	$(1 - \frac{1}{m})\frac{1}{e}(1 - \frac{1}{e})$	Theorem 4.4.2
matroid	2	$\frac{1}{10}$	Corollary 4.3.4
	2	$\frac{1-\frac{1}{m}}{2+e}$	Theorem 4.4.1
	$O\left(\frac{1}{\epsilon}\right)$	$\frac{1}{e} - \epsilon$	Theorem 5.3.3
p-system	2	$\frac{1}{2+4(p+1)}$	Corollary 4.3.4
	2	$\frac{3(1-\frac{1}{m})}{5p+7+\frac{2}{p}}$	Theorem 4.4.1

Table 1.1: Summary of results obtained for distributed submodular maximization in the constrained setting. Here $\Delta = \max_{i \in V} f(\{i\})$ and m is the number of machines. In the results of Kumar et al. [2013], in the number of rounds, Δ can be replaced by the maximum size of a solution. All algorithms in previous works and ours are randomized and the approximation guarantees stated hold in expectation.

Chapter 2

Preliminaries

In this chapter, we review definitions and the notation adopted by this thesis and reference results from literature utilized in the proofs of the subsequent chapters. We hope to provide a sufficient amount of context for our results while maintaining brevity.

Main definitions are stated in Section 2.1. In Section 2.2 we define the problems this thesis will be concerned with. Section 2.3 contains definitions and some useful results on continuous extensions for submodular functions. A description of the distributed model considered is included in Section 2.4. Finally, relevant related work, which serve as background for the results in the coming chapters, are listed in Section 2.5.

2.1 Definitions

2.1.1 Submodular functions

Given a set V, a *set function* is a function $f : 2^V \to \mathbb{R}_+$ that assigns a value to every subset of V. We refer to the set V as the *ground set*. Also, unless stated otherwise, we let $n \coloneqq |V|$ be the size of the ground set.

Submodular set functions can be defined as follows.

Definition 2.1.1. A set function $f : 2^V \to \mathbb{R}$ is *submodular* if, for all sets $A, B \subseteq V$,

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B).$$

A useful alternative characterization of submodularity can be formulated in terms of diminishing marginal gains.

Definition 2.1.2. Given a set function $f : 2^V \to \mathbb{R}$ we call the *marginal gain* of an element $e \in V$ over a set $S \subseteq V$ the difference for all $A \subseteq V$,

$$f(S \cup \{e\}) - f(S)$$

Submodular functions can then be defined equivalently as follows. Informally, it states that the marginal gain of an element e does not increase as the set grows.

Definition 2.1.3. A set function $f : 2^V \to \mathbb{R}$ is *submodular* if, for all $A \subseteq B \subseteq V$ and $e \notin B$,

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$$

Moreover, we shall consider both monotone and non-monotone submodular functions. The monotonicity property of set functions can be defined as follows.

Definition 2.1.4. A set function $f : 2^V \to \mathbb{R}$ is *monotone* if, for all $A \subseteq V$ and $e \notin A$,

$$f(A \cup \{e\}) - f(A) \geq 0$$

Since dealing with maximization problems, we will focus on *non-negative* set functions.

Definition 2.1.5. A set function $f : 2^V \to \mathbb{R}$ is *non-negative* if, for all $A \subseteq V$, $f(A) \geq 0$. Alternatively, we may represent a non-negative function f by $f : 2^V \to \mathbb{R}_+$.

Note that representing a submodular function f explicitly, that is, listing values associated with all possible subsets of the ground set V could lead to an exponential representation, which is unsuitable for our purposes. Hence, we assume in general f is represented by a *value oracle*: a black box that takes a set $S \subseteq V$ as input, and returns the value of $f(S)$.

Examples

A classical example of submodular function is the *coverage* function. Let E be a set of elements and $V \coloneqq \{S_1, S_2, \ldots, S_n\}$ a collection of subsets of E. The coverage function $f : 2^V \to \mathbb{R}_+$ associated is then given by $f(S) \coloneqq |\cup_{S_i \in S} S_i|$ (the number of elements of E that appear in any set in S), for $S \subseteq V$. This function is submodular and monotone.

Another common example of submodular functions is the *graph cut* function, that is, the function giving the number of edges crossing a cut in a graph (directed or undirected). More formally, let $G = (V, E)$ be a graph and let $w : E \to \mathbb{R}_+$ be a weight function on the edges of a graph. For a set of vertices S, let $\delta(S)$ be the set of all edges (a, b) such that $a \in S$ and $b \in V \setminus S$ (or vice-versa if G is undirected). The graph cut function $f : 2^V \to \mathbb{R}_+$ is defined by $f(S) \coloneqq \sum_{e \in \delta(S)} w(e)$. This function is submodular and non-monotone.

2.1.2 Hereditary families of sets

A *hereditary family* $\mathcal{I}$ (also called *down-monotone family* or *downward closed family*) over V ($\mathcal{I} \subseteq 2^V$) is defined as a collection of subsets of V such that if a set is in the family, so are all its subsets.

Common hereditary families include set families defined by cardinality constraints (for $k \geq 0$, the family of all subsets with at most k elements, that is, $\mathcal{I} = \{S \subseteq V : |S| \leq k\}$); and knapsack constraints (for $b \geq 0$, and weight function $w : V \to \mathbb{R}_+$, consider the family of all subsets of weight at most b, that is, $\mathcal{I} = \{S \subseteq V : \sum_{i \in S} w_i \leq b\}$). Other examples include matroids and p-systems, both defined in the following.

Definition 2.1.6. The pair $\mathcal{M} = (V, \mathcal{I})$ is a *matroid* if $\mathcal{I}$ is hereditary and it satisfies the *augmentation property*: for all A and B in $\mathcal{I}$, if $|A| < |B|$, then there is some e in $B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$.

Given a matroid $\mathcal{M} = (V, \mathcal{I})$, we call an inclusion-wise maximal set of $\mathcal{I}$ a *basis*. The following well-known lemma follows from the exchange property of a matroid (see, for example, Schrijver [2003]).

Lemma 2.1.7. *Let $\mathcal{M} = (V, \mathcal{I})$ be a matroid and let $A, B \in \mathcal{I}$ be two bases in the matroid. There is a bijection $\pi : A \to B$ such that for every element $e \in A$ we have $A \setminus \{e\} \cup \{\pi(e)\} \in \mathcal{I}$.*

Similarly to the definition of basis for matroids, given $V' \subseteq V$ and hereditary family $\mathcal{I}$, define $\mathcal{B}(V') \coloneqq \{A \in \mathcal{I} : A \subseteq V' \text{ and there is no } A' \in \mathcal{I} \text{ such that } A \subset A' \subseteq V'\}$ to be the inclusion-wise maximal sets of $\mathcal{I}$ in V'. The notion of p-systems can be defined as follows [Calinescu et al., 2007; Jenkyns, 1976; Hausmann and Korte, 1978].

Definition 2.1.8. The pair $\mathcal{P} = (V, \mathcal{I})$ is a *p-system* if $\mathcal{I}$ is hereditary and for all $V' \subseteq V$, we have $\max_{S \in \mathcal{B}(V')} |S| \leq p \cdot \min_{S \in \mathcal{B}(V')} |S|$

We note that p-systems generalize the intersection of p matroids.

2.2 Problems considered

In this thesis, we consider generally problems of the type

$$\max\{f(S) : S \subseteq V, S \in \mathcal{I}\}, \tag{2.1}$$

where $f : 2^V \to \mathcal{R}_+$ is a submodular function and $\mathcal{I}$ is any hereditary family over V (we shall also call $\mathcal{I}$ a hereditary *constraint* interchangeably in the context of optimization). We represent an instance to such problems by the triple $\langle V, \mathcal{I}, f \rangle$.

If $\mathcal{I}$ is the power set of V (that is, $\mathcal{I} = 2^V$), we call the problem *unconstrained*, as one could simply ignore the constraint $S \in \mathcal{I}$. In such cases, we will omit $\mathcal{I}$ from the problem description. If, on the other hand, $\mathcal{I} \subset 2^V$, we say the problem is *constrained*.

Whenever clear from context, we are going to refer to an optimal solution to the problem considered simply as OPT. In the text, we shall consider both monotone and non-monotone submodular functions. However, the following simple observation shows that even non-monotone submodular functions are monotone when restricted to the optimal solution of a problem of the sort we consider.

Lemma 2.2.1. *Let f be a submodular function and $\mathrm{OPT} = \arg\max_{S \in \mathcal{I}} f(S)$ for some hereditary constraint $\mathcal{I}$. Then, $f(A \cap \mathrm{OPT}) \leq f(B \cap \mathrm{OPT})$ for all $A \subseteq B$.*

Proof. Consider $X \subseteq \mathrm{OPT}$ and $e \in \mathrm{OPT} \setminus X$. By submodularity, $f(X \cup \{e\}) - f(X) \geq f(\mathrm{OPT}) - f(\mathrm{OPT} \setminus \{e\})$. On the other hand, because $\mathcal{I}$ is hereditary, $\mathrm{OPT} \setminus \{e\}$ is feasible and thus $f(\mathrm{OPT}) \geq f(\mathrm{OPT} \setminus \{e\})$. Therefore $f(X \cup \{e\}) - f(X) \geq 0$ for all X and $e \in \mathrm{OPT} \setminus X$. □

Throughout the text, we will use $k \coloneqq \max_{S \in \mathcal{I}} |S|$ as the maximum size of a solution. When dealing with distributed algorithms, we will denote by m the number of machines employed by the distributed algorithm.

2.3 Continuous extensions

Submodular functions are defined in the domain $\{0,1\}^V$. However, some of our analyses and algorithms will make use of continuous extensions, that is, functions in the domain $[0,1]^V$. In the following, we generally use $\mathbf{x}$ (or, alternatively, $\mathbf{p}$) to indicate a vector in $[0,1]^V$. We shall also make use of the definition of *characteristic vector* (or *indicator vector*) of a set. If $\mathbf{x}$ is the characteristic vector of a set $S \subseteq V$, then for every $e \in V$, we have $\mathbf{x}_e = 1$ if $e \in S$; and $\mathbf{x}_e = 0$ otherwise. Throughout the text, we denote by $\mathbf{1}_S$ the characteristic vector of a set $S \subseteq V$.

There are a few possible ways to define such extensions, but they all share a common property. Let $f : \{0,1\}^V \to \mathbb{R}_+$ be a submodular function and let $\hat{f} : [0,1]^V \to \mathbb{R}_+$ be a continuous extension of f. Then, for every set $S \subseteq V$, we have $\hat{f}(\mathbf{1}_S) = f(S)$. In

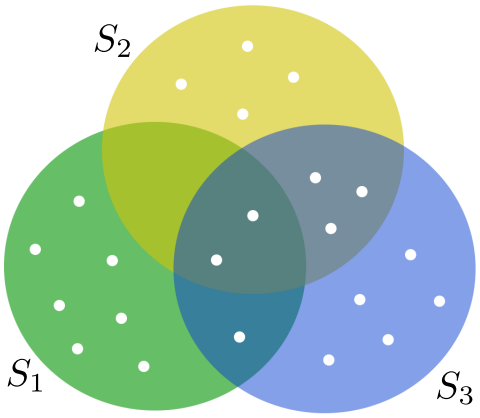

Figure 2.1: An instance of coverage function with 3 sets: $V = \{S_1, S_2, S_3\}$. The set of elements N (as described in Section 2.1.1) is represented by the white dots.

other words, the value of the extension for the characteristic vector of a set S coincides with the value of the submodular function for the same set, for every set S.

In this thesis, we work with two standard continuous extensions of submodular functions, the multilinear extension and the Lovász extension, defined in the following.

We will use the instance depicted in Figure 2.1 as a running example to the extensions introduced. The figure shows an instance of coverage problem (defined in Section 2.1.1) with 3 sets S_1, S_2, S_3, each covering a number of points.

2.3.1 Lovász extension

The *Lovász extension* [Lovász, 1983] of a submodular function f is the function $f^- : [0,1]^V \to \mathbb{R}_+$ such that $f^-(\mathbf{x}) = \mathbf{E}_{\theta \in \mathcal{U}(0,1)}[f(\{e : x_e \geq \theta\})]$, where $\mathcal{U}(0,1)$ is the uniform distribution on [0,1]. For any submodular function f, the Lovász extension f^- satisfies the following properties: (1) f^- is convex, and (2) $f^-(c \cdot \mathbf{x}) \geq c \cdot f^-(\mathbf{x})$ for any $c \in [0,1]$.

In the example given in Figure 2.1, if we consider vector $\mathbf{x} = (0.7, 0.3, 0.1)$, the value of the Lovász extension f^- would be

$$f^-(\mathbf{x}) = 0.3f(\emptyset) + 0.4f(\{S_1\}) + 0.2f(\{S_1 \cup S_2\}) + 0.1f(\{S_1 \cup S_2 \cup S_3\}) = 9.6$$

We shall make use of the following lemmas. In what follows, and throughout the text, we call generally a *random set* a set where each element is included with some probability (not necessarily independent).

Lemma 2.3.1 (da Ponte Barbosa et al. [2015])**.** *Let S be a random set with $\mathbf{E}[\mathbf{1}_S] = c \cdot \mathbf{p}$, for $c \in [0,1]$ and $\mathbf{p} \in [0,1]^V$. Then, $\mathbf{E}[f(S)] \geq c \cdot f^-(\mathbf{p})$.*

Proof. We have:

$$\mathbf{E}[f(S)] = \mathbf{E}[f^-(\mathbf{1}_S)] \geq f^-(\mathbf{E}[\mathbf{1}_S]) = f^-(c \cdot \mathbf{p}) \geq c \cdot f^-(\mathbf{p}),$$

where the first inequality follows from property (1), and the final inequality from property (2). □

Lemma 2.3.2 (da Ponte Barbosa et al. [2016])**.** *Let $f : 2^V \to \mathbb{R}_+$ be a submodular function that is monotone when restricted to $X \subseteq V$. Further, let $T, S \subseteq X$, and let R be a random subset of T in which every element occurs with probability at least p. Then, $\mathbf{E}[f(R \cup S)] \geq p \cdot f(T \cup S) + (1-p)f(S)$.*

Proof. From property (1) of the Lovász extension of f, f^- is convex. Then,

$$\mathbf{E}[f(R \cup S)] = \mathbf{E}[f^-(\mathbf{1}_{R \cup S})] \geq f^-(\mathbf{E}[\mathbf{1}_{R \cup S}]) = f^-(\mathbf{E}[\mathbf{1}_{R \setminus S}] + \mathbf{1}_S).$$

Since every element of T occurs in R with probability at least p, we have $\mathbf{E}[\mathbf{1}_{R \setminus S}] \geq p \cdot \mathbf{1}_{T \setminus S}$. Then, since f is monotone with respect to $X \supseteq S \cup T$, we must have:

$$f^-(\mathbf{E}[\mathbf{1}_{R \setminus S}] + \mathbf{1}_S) \geq f^-(p \cdot \mathbf{1}_{T \setminus S} + \mathbf{1}_S).$$

Finally, from the definition of f^-, we have

$$f^-(p \cdot \mathbf{1}_{T \setminus S} + \mathbf{1}_S) = p \cdot f(T \cup S) + (1-p)f(S).$$

□

2.3.2 Multilinear extension

The *multilinear extension* of a submodular function f is the function $F : [0,1]^V \to \mathbb{R}_+$ such that $F(\mathbf{x}) = \mathbf{E}[f(R(\mathbf{x}))]$, where $R(\mathbf{x})$ is a random subset of V in which each element e appears independently with probability $\mathbf{x}_e$. Alternatively, given a vector $\mathbf{x} \in [0,1]^V$ and submodular function f, the multilinear extension can be expressed with the following formula:

$$F(\mathbf{x}) = \sum_{S \subseteq V} f(S) \prod_{e \in S} \mathbf{x}_e \prod_{e \in V \setminus S} (1 - \mathbf{x}_e).$$

Returning to the example in Figure 2.1, and considering the same vector $\mathbf{x} =$

$(0.7, 0.3, 0.1)$ as above, we get the following value for the multilinear extension

$$\begin{aligned} F(\mathbf{x}) =& (0.7 \cdot 0.7 \cdot 0.9) f(\{S_1\}) + (0.3 \cdot 0.3 \cdot 0.9) f(\{S_2\}) + (0.3 \cdot 0.7 \cdot 0.1) f(\{S_3\}) + \\ & (0.7 \cdot 0.3 \cdot 0.9) f(\{S_1 \cup S_2\}) + (0.7 \cdot 0.7 \cdot 0.1) f(\{S_1 \cup S_3\}) + \\ & (0.3 \cdot 0.3 \cdot 0.1) f(\{S_2 \cup S_3\}) + (0.7 \cdot 0.3 \cdot 0.1) f(\{S_1 \cup S_2 \cup S_3\}) = 10.063 \end{aligned}$$

We denote by $\frac{\partial F}{\partial \mathbf{x}_i}$ the partial derivative of F with respect to $\mathbf{x}_i$, and by ∇F the gradient of F. Given vectors $\mathbf{x}$ and $\mathbf{y}$ in $[0,1]^V$, we define the vectors $\mathbf{x} \vee \mathbf{y}$ and $\mathbf{x} \wedge \mathbf{y}$ as the coordinate-wise maximum and minimum, respectively. In other words, $(\mathbf{x} \vee \mathbf{y})_e \coloneqq \max\{\mathbf{x}_e, \mathbf{y}_e\}$ and $(\mathbf{x} \wedge \mathbf{y})_e \coloneqq \min\{\mathbf{x}_e, \mathbf{y}_e\}$.

The multilinear extension has the following properties.

Claim 2.3.3. *Let $f : 2^V \to \mathbb{R}_+$ be a submodular function, and $\mathbf{x} \in [0,1]^V$. We have:*

$$\frac{\partial F}{\partial \mathbf{x}_e}(\mathbf{x}) = F(\mathbf{x} \vee \mathbf{1}_{\{e\}}) - F(\mathbf{x} \wedge \mathbf{1}_{V \setminus \{e\}}) = \frac{F(\mathbf{x} \vee \mathbf{1}_{\{e\}}) - F(\mathbf{x})}{1 - \mathbf{x}_e} = \frac{F(\mathbf{x}) - F(\mathbf{x} \wedge \mathbf{1}_{V \setminus \{e\}})}{\mathbf{x}_e}.$$

Proof. Note that, for any i, $F(\mathbf{x})$ can be written as $F(\mathbf{x}) = \mathbf{x}_i \cdot F(\mathbf{x} \vee \mathbf{1}_{\{i\}}) + (1 - \mathbf{x}_i) \cdot F(\mathbf{x} \wedge \mathbf{1}_{V \setminus \{i\}})$. The first identity is obtained by taking the partial derivative with respect to $\mathbf{x}_i$. The other two identities can be obtained using that definition of F and the partial derivative. □

For any set $S \subseteq V$, the following inequality relates the value of $F(\mathbf{1}_S \vee \mathbf{x})$ to the largest value of $\mathbf{x}$.

Lemma 2.3.4. (Feldman et al. [2011], Lemma III.5) *Let $f : 2^V \to \mathbb{R}_+$ be a non-negative submodular function, set $S \subseteq V$, and $\mathbf{x} \in [0,1]^V$. Assuming $\mathbf{x}_e \leq a$ for every $e \in V$, then $F(\mathbf{1}_S \vee \mathbf{x}) \geq (1 - a) f(S)$.*

Proof. Suppose, without loss of generality, that the elements in V are indexed in non-increasing order of values in $\mathbf{x}$. This way, $\mathbf{x}_1 \geq \mathbf{x}_2 \geq \ldots \geq \mathbf{x}_n$.

Let $\hat{\mathbf{x}}_k$ be the vector that has value $\mathbf{x}_i$ for every i such that $1 \leq i \leq k$, and 0 otherwise; we define $\hat{\mathbf{x}}_0 \coloneqq \mathbf{0}$. Let also X_i be the subset of V that contains elements 1 to i; define $X_0 \coloneqq \emptyset$.

We have

$$
\begin{aligned}
F(\mathbf{1}_S \vee \mathbf{x}) &= F(\mathbf{1}_S) + \sum_{i=1}^{n} F(\mathbf{1}_S \vee \hat{\mathbf{x}}_i) - F(\mathbf{1}_S \vee \hat{\mathbf{x}}_{i-1}) \\
&= F(\mathbf{1}_S) + \sum_{i=1}^{n} \mathbf{x}_i \left(F(\mathbf{1}_S \vee \hat{\mathbf{x}}_{i-1} \vee \mathbf{1}_{\{i\}}) - F(\mathbf{1}_S \vee \hat{\mathbf{x}}_{i-1}) \right) \quad \text{(by Claim 2.3.3)} \\
&\geq F(\mathbf{1}_S) + \sum_{i=1}^{n} \mathbf{x}_i \left(F(\mathbf{1}_S \vee \mathbf{1}_{X_{i-1}} \vee \mathbf{1}_{\{i\}}) - F(\mathbf{1}_S \vee \mathbf{1}_{X_{i-1}}) \right) \\
& \qquad \text{(by submodularity)} \\
&= f(S) + \sum_{i=1}^{n} \mathbf{x}_i \left(f(S \cup X_{i-1} \cup \{i\}) - f(S \cup X_{i-1}) \right) \\
&\geq (1 - \mathbf{x}_1) f(S) \qquad \text{(by non-negativity of } f\text{)} \\
&\geq (1 - a) f(S)
\end{aligned}
$$

□

2.4 Parallel models

2.4.1 MapReduce

In a MapReduce computation, the data is represented as $\langle \text{key}, \text{value} \rangle$ pairs and it is distributed across m machines. The computation proceeds in rounds. In a given round, the data is processed in parallel on each of the machines by *map tasks* that output $\langle \text{key}, \text{value} \rangle$ pairs. These pairs are then shuffled by *reduce tasks*; each reduce task processes all the $\langle \text{key}, \text{value} \rangle$ pairs with a given key. The output of the reduce tasks either becomes the final output of the MapReduce computation or it serves as the input of the next MapReduce round.

We adopt in Chapters 4 and 5 the most stringent MapReduce-style model among [Karloff et al., 2010; Goodrich et al., 2011; Beame et al., 2013; Andoni et al., 2014], the Massively Parallel Communication (MPC) model from Beame et al. [2013] as specified by Andoni et al. [2014]. Let N be the size of the input. In this model, there are M machines each with space S. The total memory of the system is $M \cdot S = O(N)$, which is at most a constant factor more than the input size. Computation proceeds in synchronous rounds. In each round, each machine can perform local computation and at the end, it can send at most a total of $O(S)$ words to other machines. These $O(S)$ words could form a single message of size S, S messages of size 1, or any other combination whose sum is at most $O(S)$. Following Karloff et al. [2010], we restrict both $M, S < N^{1-\Omega(1)}$. The typical

main complexity measure is the number of rounds.

Note that not all previous works on MapReduce-style algorithms for constrained submodular maximization satisfy the strict requirements of the MPC model. For instance, the previous work by Kumar et al. [2013] requires $\Theta(N \log N)$ total memory and thus it does not fit in this model (though it might be possible to modify their algorithms to satisfy this).

We assume that the size of the solution is at most N^{1-c} for some constant $0 < c < 1$. Thus, an entire solution can be stored on a single machine in the model. This assumption is also used in previous work such as [Mirrokni and Zadimoghaddam, 2015].

2.4.2 Parallel transaction processing systems

In Chapter 6, we consider the design of parallel algorithms from the perspective of *parallel transaction processing systems* [Özsu and Valduriez, 2011; Kung and Robinson, 1981; Pan et al., 2014]. In this model, there are client machines, whose task is to compute changes to be applied to the data; and a server machine, which keeps track of the current state of the data, or *program state*.

The client machines, in parallel, construct *transactions* (that is, operations to be performed to the data) under limited assumptions about the program state. After pulling the next element to be processed from the server, the client requests bounds on the program state. The transactions are constructed considering these bounds on the program state, as it may have changed due to operations committed by other clients. If the bound is insufficient to construct the transaction then a *fail* message is returned. The client then sends the proposed change to the server and proceeds to the next element.

The server, in turn, serially processes the transactions, advancing the program state. If the bounds were insufficient and the transaction failed at the client, then the server serially reconstructs and applies the transaction under the true program state.

The main objective in this model is to plan the operations so that the resulting algorithm is highly parallel, while still maintaining a solution of good quality. A high degree of parallelism in this model is attained when the cost of constructing transactions dominates the cost of applying the transactions.

2.5 Background and related work

Work on parallel and distributed algorithms for submodular maximization has been relatively limited. Early results considered the special case of maximum k-coverage, and attained an $O(1 - 1/e - \epsilon)$-approximation [Chierichetti et al., 2010; Blelloch et al.,

2011]. Later, Kumar et al. [2013] considered the more general problem of maximizing an arbitrary monotone submodular function subject to a matroid, knapsack, or p-system constraint. Their approach attains a $\frac{1}{2+\epsilon}$ approximation for matroids, and requires $O(\frac{1}{\epsilon}\log\Delta)$ MapReduce rounds, where Δ is the value of the best single element. More generally, they obtain a $\frac{1}{p+1+\epsilon}$ approximation for p-systems in $O(\frac{1}{\epsilon}\log\Delta)$ rounds. The factor of $\log\Delta$ in the number of rounds is inherent in their approach: they adapt the threshold greedy algorithm, which sequentially picks elements in $\log\Delta$ different thresholds.

In another line of work, Mirzasoleiman et al. [2013] introduced a simple, two-round distributed greedy algorithm for submodular maximization. While their algorithm is only an $O(\frac{1}{m})$-approximation in the worst case, it performs very well in practice, and attains provable constant-factor guarantees for submodular functions exhibiting certain additional structure. Finally, Mirrokni and Zadimoghaddam [2015] gave the currently-best 0.545-approximation for the cardinality constraint case using only 2 rounds of MapReduce.

In the unconstrained setting, Pan et al. [2014] considered the parallelization of submodular maximization algorithms from the perspective of parallel transaction processing systems. They proposed two methods for parallelizing the double greedy algorithm of [Buchbinder et al., 2012], and analyzed the trade-off between the approximation guarantee obtained and the level of parallelism attained.

There has also been a recent push toward obtaining fast, practical algorithms for submodular maximization problems arising in a variety of applied settings. Research in this direction has yielded a variety of techniques for speeding up the continuous greedy algorithm for monotone maximization [Badanidiyuru and Vondrák, 2014; Mirzasoleiman et al., 2015], as well as new approaches for non-monotone maximization based on insights from both the continuous greedy and double greedy algorithms [Buchbinder et al., 2014a,b].

Of particular relevance to our results is the case of maximization under a matroid constraint. Here, for monotone functions the fastest current sequential algorithm gives a $1-1/e-\epsilon$ approximation using $O(\frac{\sqrt{k}n}{\epsilon^5}\ln^2(\frac{n}{\epsilon})+\frac{k^2}{\epsilon})$ value queries. For non-monotone functions, Buchbinder et al. [2014b] give an $\frac{1+e^{-2}}{4} > 0.283$-approximation in time $O(kn\log n + Mk)$, where M is the time required to compute a perfect matching on bipartite graph with k vertices per side. They also give a simple, combinatorial 1/4-approximation in time $O(kn\log n)$. In comparison, the sequential algorithm we present

here is faster by a factor of $\Omega(k)$, at the cost of a slightly weaker $\frac{1}{2+e} > 0.211$-approximation.

Chapter 3

Sequential algorithms for submodular optimization

We review in this chapter some of the main algorithms known for submodular maximization. In the constrained setting, Section 3.1 shows the Standard Greedy algorithm, while Section 3.2 presents the Continuous Greedy. For the unconstrained case, the Random Sample algorithm and the Double Greedy are shown in Sections 3.3 and 3.4, respectively.

Throughout the text, given an algorithm Alg, we denote by $\mathsf{Alg}(N, \mathcal{I}, f)$ the output of Alg when given $\langle N \subseteq V, \mathcal{I}, f \rangle$ as input. Moreover, often the constraint $\mathcal{I}$ and f will be clear from context. In those cases, we write only $\mathsf{Alg}(N)$, for some $N \subseteq V$.

3.1 Standard Greedy

The Greedy algorithm (shown in Algorithm 1) is perhaps the most natural strategy for the problem of maximizing a submodular function in constrained setting. Given $\langle N \subseteq V, \mathcal{I}, f \rangle$, and starting with an empty set, Greedy iteratively constructs a solution $S \in \mathcal{I}$ by choosing at each step the element that the most improves the current solution while maintaining feasibility.

Although quite simple, it has been shown Greedy performs remarkably well under a variety of constraints if the submodular function to be optimized is monotone.

Theorem 3.1.1. *Let $f : 2^V \to \mathbb{R}_+$ be a monotone submodular function, and $\mathcal{I} \subseteq 2^V$ be a hereditary constraint. The* Greedy *algorithm is an α-approximation for the problem $\max_{S \in \mathcal{I}} f(S)$, where α is*

- $1 - \frac{1}{e}$ *if $\mathcal{I}$ is a cardinality constraint [Nemhauser and Wolsey, 1978];*
- 0.35 *if $\mathcal{I}$ is a knapsack constraint [Wolsey, 1982];*
- $\frac{1}{2}$ *if $\mathcal{I}$ is a matroid constraint [Nemhauser et al., 1978b]; and*
- $\frac{1}{p+1}$ *if $\mathcal{I}$ is a p-system constraint [Nemhauser et al., 1978b].*

Algorithm 1: Greedy algorithm (Greedy).

Input: $N \subseteq V, \mathcal{I}, f$

1 $S \leftarrow \emptyset$ **while** $C \neq \emptyset$ **do**
2 $C \leftarrow \{e \in N \setminus S : S \cup \{e\} \in \mathcal{I}\}$
3 $e \leftarrow \arg\max_{e \in C}\{f(S \cup \{e\}) - f(S)\}$
4 **if** $C = \emptyset$ **or** $f(S \cup \{e\}) - f(S) < 0$ **then**
5 **return** S
6 **else** $S \leftarrow S \cup \{e\}$

We provide in the following the proofs of Theorem 3.1.1 for cardinality and matroid constraints.

Proof (Cardinality constraint). Let S be the set returned by the algorithm; let $e_1, e_2, \ldots, e_k$ be the elements added to S by the Greedy algorithm in line 6 in order; and let $S_i = \{e_1, \ldots, e_i\}$ be the set S after the ith insertion (with $S_0 = \emptyset$).

Then,

$$
\begin{aligned}
f(\mathrm{OPT}) &\leq f(\mathrm{OPT} \cup S_{i-1}) && \text{(Monotonicity of } f\text{)} \\
&\leq f(S_{i-1}) + \sum_{o \in \mathrm{OPT} \setminus S_{i-1}} f(S_{i-1} \cup \{o\}) - f(S_{i-1}) && \text{(Submodularity)} \\
&\leq f(S_{i-1}) + \sum_{o \in \mathrm{OPT} \setminus S_{i-1}} f(S_i) - f(S_{i-1}) && \text{(Greedy choice of the algorithm)} \\
&\leq f(S_{i-1}) + k\left(f(S_i) - f(S_{i-1})\right) && \text{(Cardinality constraint)}
\end{aligned}
$$

Subtracting $kf(\mathrm{OPT})$ on both sides gives

$$
f(S_i) - f(\mathrm{OPT}) \geq \frac{k-1}{k}\left(f(S_{i-1}) - f(\mathrm{OPT})\right),
$$

which implies

$$
f(S_i) \geq \left(1 - \left(1 - \frac{1}{k}\right)^i\right) f(\mathrm{OPT}).
$$

Taking $i = k$ and using $(1 - 1/k)^k \leq 1/e$ gives

$$
f(S_k) = f(S) \geq \left(1 - \frac{1}{e}\right) f(\mathrm{OPT})
$$

. □

Proof (Matroid constraint). Let S be the set returned by the algorithm, and OPT an

optimal solution. From Lemma 2.1.7, there is a bijection $\pi : \text{OPT} \to S$. Let $\text{OPT} = \{o_1, o_2, \ldots, o_k\}$ and $S = \{e_1, e_2, \ldots, e_k\}$, where $e_i = \pi(o_i)$. Also, as before, let $S_i = \{e_1, \ldots, e_i\}$, and $S_0 = \emptyset$.

$$
\begin{aligned}
f(\text{OPT}) &\leq f(\text{OPT} \cup S) && \text{(Monotonicity of } f\text{)} \\
&\leq f(S) + \sum_{o_i \in \text{OPT} \setminus S} f(S \cup \{o_i\}) - f(S) && \text{(Submodularity)} \\
&\leq f(S) + \sum_{i=1}^{k} f(S \cup \{o_i\}) - f(S) && \text{(Marginal value is zero if } o_i \in S\text{)} \\
&\leq f(S) + \sum_{i=1}^{k} f(S_{i-1} \cup \{o_i\}) - f(S_{i-1}) && \text{(Submodularity)} \\
&\leq f(S) + \sum_{i=1}^{k} f(S_i) - f(S_{i-1}) \\
& && \text{(Bijection } \pi\text{, and greedy choice of the algorithm)} \\
&\leq 2f(S)
\end{aligned}
$$

Thus,

$$f(S) \geq \frac{1}{2} f(\text{OPT})$$

. □

Lastly, we remark that the Greedy algorithm will be also used here for non-monotone optimization.

3.1.1 The strong greedy property

The following property, satisfied by the Greedy algorithm for different constraints (with some γ), will be key in some of our analyses.

$$\text{For all } S \in \mathcal{I}: \qquad f(\mathsf{Greedy}(V)) \geq \gamma \cdot f(\mathsf{Greedy}(V) \cup S) \tag{GP}$$

The standard analysis of the Greedy algorithm shows that (GP) is satisfied with constant γ for many hereditary constraints, such as $\gamma = 1/2$ for a matroid constraint; and $\gamma = 1/(p+1)$ for a p-system constraint.

We show in the following the case for matroids; the proof is similar to the one for the matroid case in Theorem 3.1.1.

Lemma 3.1.2. *Let $f : 2^V \to \mathbb{R}_+$ be a submodular function, and $\mathcal{I} \subseteq 2^V$ be a matroid constraint. The* Greedy *algorithm for the problem $\max_{S \in \mathcal{I}} f(S)$ satisfies* (GP) *with $\gamma = 1/2$.*

Proof. Let $G := \mathsf{Greedy}(V)$ be the set returned by Greedy, and $S \in \mathcal{I}$. From Lemma 2.1.7, there is a bijection $\pi : S \to G$. Let $S = \{s_1, s_2, \ldots, s_k\}$ and $G = \{g_1, g_2, \ldots, g_k\}$, where $g_i = \pi(s_i)$. Also, as before, let $G_i = \{g_1, \ldots, g_i\}$, and $G_0 = \emptyset$.

$$
\begin{aligned}
f(G \cup S) &\leq f(G) + \sum_{s_i \in S \setminus G} f(G \cup \{s_i\}) - f(G) && \text{(Submodularity)} \\
&\leq f(G) + \sum_{i=1}^{k} f(G \cup \{s_i\}) - f(G) && \text{(Marginal value is zero if } s_i \in G\text{)} \\
&\leq f(G) + \sum_{i=1}^{k} f(G_{i-1} \cup \{s_i\}) - f(G_{i-1}) && \text{(Submodularity)} \\
&\leq f(G) + \sum_{i=1}^{k} f(G_i) - f(G_{i-1}) \\
& && \text{(Bijection } \pi\text{, and greedy choice of the algorithm)} \\
&\leq 2f(G)
\end{aligned}
$$

Thus,

$$f(\mathsf{Greedy}(V)) \geq \frac{1}{2} \cdot f(\mathsf{Greedy}(V) \cup S)$$

. □

3.2 Continuous Greedy

The Greedy algorithm achieves good approximation guarantees for different types of constraints, including a tight $1 - 1/e$ [Nemhauser and Wolsey, 1978; Feige, 1998] under cardinality constraint. However, the approximability under matroid constraints was left an open problem. Also, no algorithm was known for approximating non-monotone functions.

Following a common approach in optimization, Calinescu et al. [2007] propose a continuous algorithm for the problem, obtaining a $1 - 1/e$ approximation factor for optimizing a monotone function under a matroid contraint. In subsequent work, Feldman et al. [2011] show that a modification in that algorithm can also guarantee a $1/e$ approximation ratio for non-monotone functions.

Algorithm 2: Continuous Greedy algorithm (ContinuousGreedy).

Input: $N \subseteq V, \mathcal{I}, f$

$\mathbf{x}(0) \leftarrow \mathbf{0}$

for $t \leftarrow 0$ *to* 1 **do**

  $\mathbf{v}(t) \leftarrow \text{argmax}_{\mathbf{c} \in \mathcal{C}} \langle \nabla F(\mathbf{x}(t)) \bullet (\mathbf{1} - \mathbf{x}(t)), \mathbf{c} \rangle$

  Update $\mathbf{x}(t)$ according to $\frac{d\mathbf{x}(t)}{dt} = \mathbf{v}(t) \bullet (\mathbf{1} - \mathbf{x}(t))$

$S \leftarrow \text{SwapRounding}(\mathbf{x}(1), \mathcal{I})$

Return S

The algorithm works in two main stages. First, much like interior point methods, it optimizes a continuous function, specifically the multilinear extension F. Starting with the zero vector, it continuously finds a direction inside the matroid polytope using the gradient of F, and updates the current solution.

In the second stage, the fractional solution is rounded into an integral solution using a rounding algorithm such as PipageRounding [Ageev and Sviridenko, 2004] or SwapRounding [Chekuri et al., 2010]. Both rounding algorithms have been shown to be able to find an integral solution whose value is at least as good as the fractional one [Calinescu et al., 2007; Chekuri et al., 2010].

Algorithm 2 shows a high-level, continuous version of the algorithm proposed in [Feldman et al., 2011]. Discretization of the timesteps for t is not included for the sake of ease of exposition, but can be done at a small loss in the approximation factor. For further details, we refer the reader to [Calinescu et al., 2007; Feldman et al., 2011].

In the algorithm, we let $\mathcal{C}$ be the matroid polytope induced by $\mathcal{I}$. The $\bullet$ operator indicates coordinate-wise vector multiplication (also called *Hadamard product*).

Theorem 3.2.1 (Calinescu et al. [2007]; Feldman et al. [2011])**.** *Let $f : 2^V \to \mathbb{R}_+$ be a submodular function, and $\mathcal{I} \subseteq 2^V$ be a matroid. The algorithm is an α-approximation for the problem $\max_{S \in \mathcal{I}} f(S)$, where α is $(1-1/e)$ for monotone f and $1/e$ for general f.*

Theorem 3.2.1 follows immediately from Lemma 3.2.2 by using one of the aforementioned rounding methods.

Lemma 3.2.2 (Calinescu et al. [2007]; Feldman et al. [2011])**.** *Let $f : 2^V \to \mathbb{R}_+$ be a submodular function, let F be its multilinear extension, and let $\mathcal{I} \subseteq 2^V$ be a matroid. Consider the execution of* ContinuousGreedy*. The algorithm obtains $F(\mathbf{x}^{(1)}) \geq \alpha f(\text{OPT})$, where α is $(1 - 1/e)$ for monotone f and $1/e$ for general f.*

Proof. By the chain rule, for every $t \in [0, 1]$, we have

$$\begin{aligned}
\frac{dF(\mathbf{x}(t))}{dt} &= \langle \nabla F(\mathbf{x}(t)), \frac{d\mathbf{x}(t)}{dt} \rangle \\
&= \langle \nabla F(\mathbf{x}(t)), \mathbf{v}(t) \bullet (\mathbf{1} - \mathbf{x}(t)) \rangle \\
&= \langle \nabla F(\mathbf{x}(t)) \bullet (\mathbf{1} - \mathbf{x}(t)), \mathbf{v}(t) \rangle \\
&\geq \langle \nabla F(\mathbf{x}(t)) \bullet (\mathbf{1} - \mathbf{x}(t)), \mathbf{1}_{\mathrm{OPT}} \rangle && \text{(since } \mathbf{1}_{\mathrm{OPT}} \in \mathcal{C}\text{)} \\
&= \sum_{i \in \mathrm{OPT}} F(\mathbf{x}(t) \vee \mathbf{1}_{\{i\}}) - F(\mathbf{x}(t)) && \text{(by Claim 2.3.3)} \\
&\geq F(\mathbf{x}(t) \vee \mathbf{1}_{\mathrm{OPT}}) - F(\mathbf{x}(t)) && \text{(by submodularity)}
\end{aligned}$$

If f is monotone, $F(\mathbf{x}(t) \vee \mathbf{1}_{\mathrm{OPT}}) \geq F(\mathbf{1}_{\mathrm{OPT}})$ and we get $\frac{dF(\mathbf{x}(t))}{dt} \geq F(\mathbf{1}_{\mathrm{OPT}}) - F(\mathbf{x}(t))$. By solving the differential inequality, we obtain

$$F(\mathbf{x}(t)) \geq \left(1 - \frac{1}{e^t}\right) F(\mathbf{1}_{\mathrm{OPT}}).$$

If f is non-monotone, we apply Lemma 2.3.4. To bound $\mathbf{x}(t)$, note that it is updated according to $d\mathbf{x}(t)/dt = \mathbf{v}(t) \bullet (1 - \mathbf{x}(t)) \leq \mathbf{1} \bullet (1 - \mathbf{x}(t))$. By solving the differential inequality with $\mathbf{x}(0) = \mathbf{0}$, we get $\|\mathbf{x}(t)\|_\infty \leq \left(1 - e^{-t}\right)$. Thus, we have $\frac{dF(\mathbf{x}(t))}{dt} \geq e^{-t} F(\mathbf{1}_{\mathrm{OPT}}) - F(\mathbf{x}(t))$. Now solving this differential inequality, we obtain

$$F(\mathbf{x}(t)) \geq t \left(\frac{1}{e^t}\right) F(\mathbf{1}_{\mathrm{OPT}}).$$

The lemma follows for $\mathbf{x}(1)$ since $F(\mathbf{1}_{\mathrm{OPT}}) = f(\mathrm{OPT})$.

□

3.3 Random Sample

In this section, we analyze the Random Sample algorithm for the unconstrained case ($\max_{S \subseteq V} f(S)$) as in Feige et al. [2011], where the function f we optimize over is not necessarily monotone. The algorithm (shown in Algorithm 3) is quite simple, and works by picking each element independently with probability 1/2.

Lemma 3.3.1. *Let $f : 2^V \to \mathbb{R}_+$ be a submodular function. Let $A \subseteq V$ and $\mathbf{p}$ be a vector in $[0,1]^V$ such that $p_e = 0$ for all $e \in V \setminus A$. Let $A(\mathbf{p})$ be a random subset of A where each element $e \in A$ appears with probability exactly p_e (not necessarily independently). Then $\mathbf{E}[f(A(\mathbf{p}))] \geq f^-(\mathbf{p})$, where f^- is the Lovász extension of f.*

Algorithm 3: Random Sample algorithm (RandomSample).

Input: V, f

$A \leftarrow \emptyset$

for $i \leftarrow 1$ *to* n **do**

  with probability $1/2$

  $A \leftarrow A \cup \{i\}$

Return A

Proof. We have

$$\begin{aligned}\mathbf{E}[f(A(\mathbf{p}))] &= \mathbf{E}[f^-(\mathbf{1}_{A(\mathbf{p})})] \\ &\geq f^-\left(\mathbf{E}[\mathbf{1}_{A(\mathbf{p})}]\right) \qquad \text{(Since } f^- \text{ is convex)} \\ &= f^-(\mathbf{p})\end{aligned}$$

□

Corollary 3.3.2. *Let $f : 2^V \to \mathbb{R}_+$ be a submodular function. Let $A \subseteq V$ and $p \in [0,1]$. Let $A(p)$ be the subset of A where each element of A is included independently at random with probability p. We have*

$$\mathbf{E}[f(A(p))] \geq (1-p)f(\emptyset) + pf(A)$$

We can apply the corollary above twice to obtain the following lemma.

Lemma 3.3.3 (Feige et al. [2011])**.** *Let $f : 2^V \to \mathbb{R}_+$ be a submodular function. Let $A, B \subseteq V$ and $p, q \in [0,1]$. Let $A(p)$ and $B(q)$ be the subsets of A and B where each element is included independently with probability p and q, respectively. We have*

$$\mathbf{E}[f(A(p) \cup B(q))] \geq (1-p)(1-q)f(\emptyset) + p(1-q)f(A) + (1-p)qf(B) + pqf(A \cup B)$$

Proof. Condition on the event that $B(q) = R$. Let $g(S) = f(S \cup R)$ for each set $S \subseteq V$. Note that g is submodular and thus the corollary above gives us that

$$\mathbf{E}[g(A(p))] \geq (1-p)f(R) + pf(A \cup R)$$

Let $h(S) = f(S \cup A)$. As before, h is submodular and the corollary above gives us that

$$\mathbf{E}[h(B(q))] \geq (1-q)f(A) + qf(A \cup B)$$

By removing the conditioning, we obtain

$$\begin{aligned}\mathbf{E}[f(A(p) \cup B(q))] &\geq (1-p)\,\mathbf{E}[f(B(q))] + p\,\mathbf{E}[f(A \cup B(q))] \\ &\geq (1-p)((1-q)f(\emptyset) + qf(B)) + p((1-q)f(A) + qf(A \cup B)) \\ &= (1-p)(1-q)f(\emptyset) + p(1-q)f(A) + (1-p)qf(B) + pqf(A \cup B)\end{aligned}$$

□

Theorem 3.3.4 (Feige et al. [2011])**.** *Let $f : 2^V \to \mathbb{R}_+$ be a submodular function. The* RandomSample *algorithm is a $1/4$-approximation for the problem $\max_{S \subseteq V} f(S)$.*

Proof. Using Lemma 3.3.3, we can analyze the random sample algorithm as follows. Let R be the random sample. Note that $R \cap \mathrm{OPT}$ is a $1/2$ sample of OPT and likewise $R \cap (V \setminus \mathrm{OPT})$ is a $1/2$ sample of $V \setminus \mathrm{OPT}$. Thus we can apply the previous lemma with $p = q = 1/2$ and $A = \mathrm{OPT}$ and $B = V \setminus \mathrm{OPT}$ and obtain

$$\begin{aligned}\mathbf{E}[f(R)] &\geq \frac{1}{4}(f(\emptyset) + f(\mathrm{OPT}) + f(V \setminus \mathrm{OPT}) + f(V)) \\ &\geq \frac{1}{4}f(\mathrm{OPT})\end{aligned}$$

□

3.4 Double Greedy

We present in this section the Double Greedy algorithm and the analysis of Buchbinder et al. [2012] for the unconstrained problem $(\max_{S \subseteq V} f(S))$. It achieves an approximation guarantee of $1/2$, which matches known hardness results [Feige et al., 2011]. For this algorithm, as well as in Section 3.3, monotonicity is not assumed.

The algorithm maintains two sets, A and B, initialized to $\emptyset$ and V respectively; and the invariant that $A \subseteq B$. It proceeds to compare, for each element i in V, the marginal gain a_i of including i in A, and the marginal gain b_i of removing i from B. It then takes one of the two following choices randomly: with probability proportional to a_i, element i is included in A; with probability proportional to b_i, element i is removed from B. In the algorithm description, we let $[x]_+ = \max\{x, 0\}$.

Theorem 3.4.1 (Buchbinder et al. [2012])**.** *Let $f : 2^V \to \mathbb{R}_+$ be a submodular function. The* DoubleGreedy *algorithm is a $1/2$-approximation for the problem $\max_{S \subseteq V} f(S)$.*

Algorithm 4: Double Greedy algorithm (DoubleGreedy).

Input: V, f

$A_0 \leftarrow \emptyset$
$B_0 \leftarrow V$
for $i \leftarrow 1$ *to* n **do**
  $a_i \leftarrow f(A_{i-1} \cup \{i\}) - f(A_{i-1})$
  $b_i \leftarrow f(B_{i-1} \setminus \{i\}) - f(B_{i-1})$
  $p_i \leftarrow [a_i]_+ / ([a_i]_+ + [b_i]_+)$
  with probability p_i $\langle\langle$*or if* $[a_i]_+ = [b_i]_+ = 0\rangle\rangle$
    $A_i \leftarrow A_{i-1} \cup \{i\}$
    $B_i \leftarrow B_{i-1}$
  else $\langle\langle$*with probability* $1 - p_i\rangle\rangle$
    $A_i \leftarrow A_{i-1}$
    $B_i \leftarrow B_{i-1} \setminus \{i\}$
Return A_n

The following lemma is the core of the analysis, and will also prove useful later in this thesis. If OPT be an optimal solution, for any step i in the DoubleGreedy algorithm, let $\mathrm{OPT}_i \coloneqq (\mathrm{OPT} \cup A_i) \cap B_i$, that is, the solution that agrees with the decisions made by the algorithm so far, and it agrees with OPT on the elements that are still undecided. In the analysis, the expectation is over the random choices made by the algorithm. For each step i, we will analyze how much $\mathbf{E}[f(A_i) + f(B_i)]$ increases and how much the $\mathbf{E}[f(\mathrm{OPT}_i)]$ decreases.

Note that, by the definition, $\mathrm{OPT}_0 = \mathrm{OPT}$ and $\mathrm{OPT}_n = A_n = B_n$, and thus, Theorem 3.4.1 follows from the Lemma by summing the inequality given for all i from 1 to n.

Lemma 3.4.2. *Let A_i and B_i, for i in $\{1, \ldots, n\}$, be the sets constructed by* DoubleGreedy. *We have*

$$\frac{1}{2}\,\mathbf{E}[f(A_i) + f(B_i) - f(A_{i-1}) - f(B_{i-1})] \geq \mathbf{E}[f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i)].$$

Proof. Let $a'_i \coloneqq [a_i]_+$ and $b'_i \coloneqq [b_i]_+$. First, suppose $a'_i + b'_i > 0$. We have:

$$\mathbf{E}[f(A_i) - f(A_{i-1})] = \frac{a'_i}{a'_i + b'_i}\,(f(A_{i-1} \cup \{i\}) - f(A_{i-1})) = \frac{(a'_i)^2}{a'_i + b'_i}$$

and

$$\mathbf{E}[f(B_i) - f(B_{i-1})] = \frac{b'_i}{a'_i + b'_i}\,(f(B_{i-1} \setminus \{i\}) - f(B_{i-1})) = \frac{(b'_i)^2}{a'_i + b'_i}.$$

Now, let us bound $\mathbf{E}[f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i)]$. If $i \notin \mathrm{OPT}$, we have $\mathrm{OPT}_{i-1} \subseteq$

$B_{i-1} \setminus \{i\}$. In this case, submodularity gives that $b_i \geq f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1} \cup \{i\})$, and we get

$$\begin{aligned}\mathbf{E}[f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i)] &= \frac{a'_i}{a'_i + b'_i}(f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1} \cup \{i\})) \\ &\leq \frac{a'_i b_i}{a'_i + b'_i} \\ &\leq \frac{a'_i b'_i}{a'_i + b'_i}\end{aligned}$$

On the other hand, if $i \in \mathrm{OPT}$, we have $A_{i-1} \subseteq \mathrm{OPT}_{i-1} \setminus \{i\}$, since $i \notin A_{i-1}$. It follows from submodularity that $a_i \geq f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1} \setminus \{i\})$, and we get

$$\begin{aligned}\mathbf{E}[f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i)] &= \frac{b'_i}{a'_i + b'_i}(f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_{i-1} \setminus \{i\})) \\ &\leq \frac{b'_i a_i}{a'_i + b'_i} \\ &\leq \frac{a'_i b'_i}{a'_i + b'_i}\end{aligned}$$

Since $2a'_i b'_i \leq (a'_i)^2 + (b'_i)^2$, the lemma follows.

Now suppose $a'_i + b'_i = 0$. Notice that, by submodularity, $a_i \geq -b_i$. Thus, it must be the case that $a_i = b_i = 0$, and $f(A_i) = f(A_{i-1})$ and $f(B_i) = f(B_{i-1})$. If $i \in \mathrm{OPT}$, we have trivially that $f(\mathrm{OPT}_i) = f(\mathrm{OPT})$. Otherwise, if $i \notin \mathrm{OPT}$, submodularity implies (as above)

$$f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i) \leq f(B_{i-1} \setminus \{i\}) - f(B_{i-1}) = b_i = 0$$

In both cases, the result follows. □

Chapter 4

Two-round distributed algorithms for constrained submodular maximization

Mirzasoleiman et al. [2013] give a distributed algorithm, called GreeDI, for maximizing a monotone submodular function subject to a cardinality constraint. The GreeDI algorithm partitions the data arbitrarily on the machines and on each machine it runs the classical Greedy algorithm to select a feasible subset of the items on that machine. The Greedy solutions on these machines are then placed on a single machine and the Greedy algorithm is used once more to select the final solution. The GreeDI algorithm is very simple and embarrassingly parallel, but its worst-case approximation guarantee[1] is $1/\Theta\left(\min\left\{\sqrt{k}, m\right\}\right)$, where m is the number of machines and k is the cardinality constraint. Despite this, the authors show that the GreeDI algorithm achieves very good approximations for datasets with geometric structure.

In this chapter, we show that we can achieve both the communication efficiency of the GreeDI algorithm and a provable, constant factor, approximation guarantee. Our algorithm is in fact the GreeDI algorithm with a very simple and crucial modification: instead of partitioning the data arbitrarily on the machines, we *randomly* partition the dataset. Our analysis provides some theoretical justification for the very good empirical performance of the GreeDI algorithm that was established previously in the extensive experiments of Mirzasoleiman et al. [2013]. It also suggests the approach can deliver good performance in much wider settings than originally envisioned.

Moreover, in contrast with the work of Mirzasoleiman et al. [2013], our analysis holds for any hereditary constraint. Specifically, we show that our randomized variant

[1] Mirzasoleiman et al. [2013] give a family of instances where the approximation achieved is only $1/\min\{k, m\}$ if the solution picked on each of the machines is the optimal solution for the set of items on the machine. These instances are not hard for the GreeDI algorithm. We show in Appendix A that the GreeDI algorithm achieves an $\left(1/\Theta\left(\min\left\{\sqrt{k}, m\right\}\right)\right)$ approximation.

Constraint	RandGreeDi for monotone $\left(\frac{\alpha}{2}\right)$	NMRandGreeDi for non-monotone $\left(\frac{\gamma}{4+2\gamma}\right)$	RandGreeDi for non-monotone $\left(\left(1-\frac{1}{m}\right)\frac{\beta\gamma}{\beta+\gamma}\right)$
Cardinality	≈ 0.316	$\frac{1}{10}$	$\left(1-\frac{1}{m}\right)\frac{1}{e}\left(1-\frac{1}{e}\right)$
Matroid	$\frac{1}{4}$	$\frac{1}{10}$	$\left(1-\frac{1}{m}\right)\frac{1}{2+e}$
p-system	$\frac{1}{2(p+1)}$	$\frac{1}{2+4(p+1)}$	$3\left(1-\frac{1}{m}\right)/\left(5p+7+\frac{2}{p}\right)$

Table 4.1: Approximation results obtained for randomized two-round distributed algorithms for constrained monotone and non-monotone submodular maximization. In the approximation ratios obtained by our two-round algorithms, α is the approximation ratio of Greedy; γ is the constant with which Greedy satisfies (GP) (Section 3.1.1); and β is the approximation ratio obtained by any algorithm for $\max_{S\in\mathcal{I}} f(S)$.

of the GreeDI algorithm achieves a constant factor approximation for any hereditary, constrained problem for which the classical (centralized) Greedy algorithm achieves a constant factor approximation. This is the case not only for cardinality constraints, but also for matroid constraints, knapsack constraints, and p-system constraints [Jenkyns, 1976], which generalize the intersection of p matroid constraints. Table 4.1 summarizes the constant approximation factor obtained by our randomized GreeDI algorithm.

Additionally, we show that if the greedy algorithm satisfies a slightly stronger technical condition, the strong greedy property, defined in Section 3.1.1, then our approach gives constant factor approximations for constrained *non-monotone* submodular maximization. This is indeed the case for all of the aforementioned specific classes of problems. The resulting approximation ratios for non-monotone maximization problems are given in the two last columns of Table 4.1.

First, we show that a simple modification to RandGreeDi yields an algorithm, NMRandGreeDi, that achieves constant factor approximations for non-monotone submodular maximization. It works by running Greedy twice, instead of only once, in the first round, and sending both solutions to the last machine.

Further, we show that show that, by employing algorithms other than Greedy in the second round, the same RandGreeDi algorithm can attain even better approximation guarantees for non-monotone submodular maximization under hereditary constraints.

Finally, we show that by simulating the machines in the distributed algorithm, we also obtain a fast, sequential algorithm for maximizing a non-monotone submodular function subject to a matroid constraint. Our algorithm shows that one can preprocess the instance in $O(\frac{n}{\epsilon}\log n)$ time and obtain a set X of size $O(k/\epsilon)$ so that *it suffices to solve the problem on* X.

4.1 The Greedy consistency property

Before describing our general algorithm, recall the standard greedy algorithm, Greedy, shown in Section 3.1 (Algorithm 1). Given an instance $\langle N \subseteq V, \mathcal{I}, f \rangle$, Greedy iteratively constructs a solution $S \in \mathcal{I}$ by choosing at each step the element maximizing the marginal increase of f.

The Greedy algorithm satisfies the following property, which will be key in our analysis:

Lemma 4.1.1. *Let $A \subseteq V$ and $B \subseteq V$ be two disjoint subsets of V. Suppose that, for each element $e \in B$, we have $\mathsf{Greedy}(A \cup \{e\}) = \mathsf{Greedy}(A)$. Then $\mathsf{Greedy}(A \cup B) = \mathsf{Greedy}(A)$.*

Proof. Suppose for contradiction that $\mathsf{Greedy}(A \cup B) \neq \mathsf{Greedy}(A)$. We first note that, if $\mathsf{Greedy}(A \cup B) \subseteq A$, then $\mathsf{Greedy}(A \cup B) = \mathsf{Greedy}(A)$; this follows from the fact that each iteration of the Greedy algorithm chooses the element with the highest marginal value whose addition to the current solution maintains feasibility for $\mathcal{I}$. Therefore, if $\mathsf{Greedy}(A \cup B) \neq \mathsf{Greedy}(A)$, the former solution contains an element of B. Let e be the first element of B that is selected by Greedy on the input $A \cup B$. Then Greedy will also select e on the input $A \cup \{e\}$, which contradicts the fact that $\mathsf{Greedy}(A \cup \{e\}) = \mathsf{Greedy}(A)$. □

Informally, this simply means that if Greedy rejects some element e when presented with input $A \cup \{e\}$, then adding other similarly rejected elements to $A \cup \{e\}$ cannot cause e to be accepted.

4.2 A distributed greedy algorithm for monotone submodular maximization

4.2.1 Algorithm

We now describe our general, randomized distributed algorithm, RandGreeDi, shown in Algorithm 5. Our algorithm runs in two rounds. In the first round, we *randomly* distribute the elements of the ground set V to the machines, assigning each element to a machine chosen independently and uniformly at random. On each machine i, we execute $\mathsf{Greedy}(V_i)$ to select a feasible subset S_i of the elements on that machine. In the second round, we place all of these selected subsets on a single machine, and run some algorithm Alg on this machine in order to select a final solution T. We return whichever is better: the final solution T or the best solution amongst all the S_i from the first phase.

Algorithm 5: The distributed algorithm RandGreeDi

Input: instance $\langle V, \mathcal{I}, f \rangle$, number m of machines
for $e \in V$ **do**
  Assign e to a machine i chosen uniformly at random
Let V_i be the elements assigned to machine i
Run $\mathsf{Greedy}(V_i)$ on each machine i to obtain S_i
Place $S = \bigcup_i S_i$ on machine 1
Run $\mathsf{Alg}(S)$ on machine 1 to obtain T
Let $S' = \arg\max_i \{f(S_i)\}$
return $\arg\max\{f(T), f(S')\}$

4.2.2 Analysis

In the RandGreeDi algorithm, after the first round, each machine needs to communicate the solution obtained, which has size $O(k)$ each. Thus, the total amount that all machines communicate is $O(mk)$.

In the first round, the space used on a given machine is $O(n/m)$ in expectation[2], as the n elements are split uniformly at random among m machines. In the second round, the last machine needs space $O(n/m + mk)$ to store all the solutions.

We devote the rest of this section to the analysis of the approximation guarantee attained by the RandGreeDi algorithm. Fix $\langle V, \mathcal{I}, f \rangle$, where $\mathcal{I} \subseteq 2^V$ is a hereditary constraint, and $f : 2^V \to \mathcal{R}_+$ is any non-negative, monotone submodular function. Suppose that Greedy is an α-approximation and Alg is a β-approximation for the associated constrained monotone submodular maximization problem of the form (2.1).

Let $\mathcal{V}(1/m)$ denote the distribution over random subsets of V where each element is included independently with probability $1/m$. Let $\mathbf{p} \in [0,1]^n$ be the following vector. For each element $e \in V$, we have

$$p_e = \begin{cases} \Pr_{A \sim \mathcal{V}(1/m)}[e \in \mathsf{Greedy}(A \cup \{e\})] & \text{if } e \in \mathrm{OPT} \\ 0 & \text{otherwise} \end{cases}$$

We emphasize the above probability vector is a central notion in this work, and will be used repeatedly throughout the text.

Our main theorem follows from the next two lemmas, which characterize the quality of the best solution from the first round and that of the solution from the second round, respectively. Recall that f^- is the Lovász extension of f.

[2]One can also show this holds with high probability using Chernoff bounds.

Lemma 4.2.1. *For each machine i, $\mathbf{E}[f(S_i)] \geq \alpha \cdot f^{-}\left(\mathbf{1}_{\mathrm{OPT}} - \mathbf{p}\right)$.*

Proof. Consider machine i. Let V_i denote the set of elements assigned to machine i in the first round. Let $O_i = \{e \in \mathrm{OPT} \colon e \notin \mathsf{Greedy}(V_i \cup \{e\})\}$. We make the following key observations.

We apply Lemma 4.1.1 with $A = V_i$ and $B = O_i \setminus V_i$ to obtain that $\mathsf{Greedy}(V_i) = \mathsf{Greedy}(V_i \cup O_i) = S_i$. Since $\mathrm{OPT} \in \mathcal{I}$ and $\mathcal{I}$ is hereditary, we must have $O_i \in \mathcal{I}$ as well. Since Greedy is an α-approximation, it follows that

$$f(S_i) \geq \alpha \cdot f(O_i).$$

Since the distribution of V_i is the same as $\mathcal{V}(1/m)$, for each element $e \in \mathrm{OPT}$, we have

$$\begin{aligned} \Pr[e \in O_i] &= 1 - \Pr[e \notin O_i] = 1 - p_e \\ \mathbf{E}[\mathbf{1}_{O_i}] &= \mathbf{1}_{\mathrm{OPT}} - \mathbf{p}. \end{aligned}$$

By combining these observations with Lemma 2.3.1, we obtain

$$\mathbf{E}[f(S_i)] \geq \alpha \cdot \mathbf{E}[f(O_i)] \geq \alpha \cdot f^{-}\left(\mathbf{1}_{\mathrm{OPT}} - \mathbf{p}\right).$$

□

Lemma 4.2.2. $\mathbf{E}[f(\mathsf{Alg}(S))] \geq \beta \cdot f^{-}(\mathbf{p})$.

Proof. Recall that $S = \bigcup_i \mathsf{Greedy}(V_i)$. Since $\mathrm{OPT} \in \mathcal{I}$ and $\mathcal{I}$ is hereditary, $S \cap \mathrm{OPT} \in \mathcal{I}$. Since Alg is a β-approximation, we have

$$f(\mathsf{Alg}(S)) \geq \beta \cdot f(S \cap \mathrm{OPT}). \tag{4.1}$$

Consider an element $e \in \mathrm{OPT}$. For each machine i, we have

$$\begin{aligned} \Pr[e \in S \mid e \text{ is assigned to machine } i] &= \Pr[e \in \mathsf{Greedy}(V_i) \mid e \in V_i] \\ &= \Pr_{A \sim \mathcal{V}(1/m)}[e \in \mathsf{Greedy}(A) \mid e \in A] \\ &= \Pr_{B \sim \mathcal{V}(1/m)}[e \in \mathsf{Greedy}(B \cup \{e\})] \\ &= p_e. \end{aligned}$$

The first equality follows from the fact that e is included in S if and only if it is included in $\mathsf{Greedy}(V_i)$. The second equality follows from the fact that the distribution of V_i is identical to $\mathcal{V}(1/m)$. The third equality follows from the fact that the distribution of $A \sim \mathcal{V}(1/m)$ conditioned on $e \in A$ is identical to the distribution of $B \cup \{e\}$ where $B \sim \mathcal{V}(1/m)$. Therefore

$$\begin{aligned} \Pr[e \in S \cap \mathrm{OPT}] &= p_e \\ \mathbf{E}[\mathbf{1}_{S \cap \mathrm{OPT}}] &= \mathbf{p}. \end{aligned} \tag{4.2}$$

By combining (4.1), (4.2), and Lemma 2.3.1, we obtain

$$\mathbf{E}[f(\mathsf{Alg}(S))] \geq \beta \cdot \mathbf{E}[f(S \cap \mathrm{OPT})] \geq \beta \cdot f^-(\mathbf{p}).$$

□

Combining Lemma 4.2.2 and Lemma 4.2.1 gives us our main theorem.

Theorem 4.2.3. *Let $f : 2^V \to \mathbb{R}_+$ be a monotone submodular function, let $\mathcal{I} \subseteq 2^V$ be a hereditary set system, and let Alg be β-approximation algorithm for $\max_{S \in \mathcal{I}} f(S)$. The algorithm $\mathsf{RandGreeDi}$ is (in expectation) a $\frac{\alpha\beta}{\alpha+\beta}$-approximation for the same problem, where α is the approximation ratio of Greedy algorithm.*

Proof. Let $S_i = \mathsf{Greedy}(V_i)$, $S = \bigcup_i S_i$ be the set of elements on the last machine, and $T = \mathsf{Alg}(S)$ be the solution produced on the last machine. Then, the output D produced by $\mathsf{RandGreeDi}$ satisfies $f(D) \geq \max_i(f(S_i))$ and $f(D) \geq f(T)$. Thus, from Lemmas 4.2.1 and 4.2.2 we have:

$$\mathbf{E}[f(D)] \geq \alpha \cdot f^-(\mathbf{1}_{\mathrm{OPT}} - \mathbf{p}) \tag{4.3}$$

$$\mathbf{E}[f(D)] \geq \beta \cdot f^-(\mathbf{p}). \tag{4.4}$$

By combining (4.3) and (4.4), we obtain

$$\begin{aligned} (\beta + \alpha)\,\mathbf{E}[f(D)] &\geq \alpha\beta\big(f^-(\mathbf{p}) + f^-(\mathbf{1}_{\mathrm{OPT}} - \mathbf{p})\big) \\ &\geq \alpha\beta \cdot f^-(\mathbf{1}_{\mathrm{OPT}}) \\ &= \alpha\beta \cdot f(\mathrm{OPT}). \end{aligned}$$

In the second inequality, we have used the fact that f^- is convex and $f^-(c \cdot \mathbf{x}) \geq c f^-(\mathbf{x})$

Algorithm 6: The distributed algorithm NMRandGreeDi

Input: instance $\langle V, \mathcal{I}, f \rangle$, number m of machines

for $e \in V$ **do**
  Assign e to a machine i chosen uniformly at random
Let V_i be the elements assigned to machine i
Run Greedy(V_i) on each machine i to obtain S_i^1
Run Greedy$(V_i \setminus S_i^1)$ on each machine i to obtain S_i^2
Place $S = \bigcup_i (S_i^1 \cup S_i^2)$ on machine 1
Run Alg(S) on machine 1 to obtain T
Let $S' = \arg\max_i \{f(S_i^1), f(S_i^2)\}$
return $\arg\max\{f(T), f(S')\}$

for any constant $c \in [0, 1]$. □

If we use the standard greedy algorithm for Alg, we obtain the following simplified corollary of Theorem 4.2.3.

Corollary 4.2.4. *Let $f : 2^V \to \mathbb{R}_+$ be a monotone submodular function, and let $\mathcal{I} \subseteq 2^V$ be a hereditary set system. Suppose we use* Greedy *as* Alg*. The resulting algorithm* RandGreeDi *is (in expectation) a $\frac{\alpha}{2}$-approximation for $\max_{S \in \mathcal{I}} f(S)$, where α is the approximation ratio of* Greedy *algorithm.*

4.3 A distributed greedy algorithm for non-monotone submodular maximization

We consider the problem of maximizing a *non-monotone* submodular function subject to a hereditary constraint. Our approach is a slight modification of the randomized, distributed greedy algorithm described in Section 4.2, and it builds on the work of Gupta et al. [2010]. Again, we show how to combine the standard Greedy algorithm, together with any algorithm Alg for the non-monotone case in order to obtain a randomized, distributed algorithm for the non-monotone submodular maximization.

4.3.1 Algorithm

Our modified algorithm, NMRandGreeDi (Algorithm 6), works as follows. As in the monotone case, in the first round we distribute the elements of V uniformly at random amongst the m machines. Then, we run the standard greedy algorithm *twice* to obtain two disjoint solutions S_i^1 and S_i^2 on each machine. Specifically, each machine first runs Greedy on V_i to obtain a solution S_i^1, then runs Greedy on $V_i \setminus S_i^1$ to obtain a disjoint solution S_i^2. In the second round, both of these solutions are sent to a single machine,

which runs Alg on $S = \bigcup_i (S_i^1 \cup S_i^2)$ to produce a solution T. The best solution amongst T and all of the solutions S_i^1 and S_i^2 is then returned.

4.3.2 Analysis

We devote the rest of this section to the analysis of the algorithm. The space and communication aspects are similar to those required by RandGreeDi, analyzed in Subsection 4.2.2, and thus omitted here.

Recall the strong greedy property, defined in Section 3.1.1. Given an instance $\langle V, \mathcal{I}, f \rangle$ of non-negative submodular maximization, we assume the Greedy algorithm has the following property:

$$\text{For all } S \in \mathcal{I}: \qquad f(\mathsf{Greedy}(V)) \geq \gamma \cdot f(\mathsf{Greedy}(V) \cup S) \tag{GP}$$

where the values of γ for different hereditary constraints are given in that section.

The analysis is similar to the approach from the previous section. We define $\mathcal{V}(1/m)$ as before. We modify the definition of the vector $\mathbf{p}$ in Section 4.2.2 as follows. For each element $e \in V$, we have

$$p_e = \begin{cases} \Pr\limits_{A \sim \mathcal{V}(1/m)} \Big[e \in \mathsf{Greedy}(A \cup \{e\}) \textbf{ or } \\ \qquad\qquad e \in \mathsf{Greedy}((A \cup \{e\}) \setminus \mathsf{Greedy}(A \cup \{e\})) \Big] & \text{if } e \in \mathrm{OPT} \\ 0 & \text{otherwise} \end{cases}$$

We now derive analogues of Lemmas 4.2.1 and 4.2.2.

Lemma 4.3.1. *Suppose that* Greedy *satisfies* (GP). *For each machine i,*

$$\mathbf{E}\left[f(S_i^1) + f(S_i^2)\right] \geq \gamma \cdot f^-(\mathbf{1}_{\mathrm{OPT}} - \mathbf{p}),$$

and therefore

$$\mathbf{E}\left[\max\left\{f(S_i^1), f(S_i^2)\right\}\right] \geq \frac{\gamma}{2} \cdot f^-(\mathbf{1}_{\mathrm{OPT}} - \mathbf{p}).$$

Proof. Consider machine i and let V_i be the set of elements assigned to machine i in the first round. Let

$$O_i = \{e \in \mathrm{OPT} \colon e \notin \mathsf{Greedy}(V_i \cup \{e\}) \textbf{ and } \\ e \notin \mathsf{Greedy}((V_i \cup \{e\}) \setminus \mathsf{Greedy}(V_i \cup \{e\}))\}$$

Note that, since $\mathrm{OPT} \in \mathcal{I}$ and $\mathcal{I}$ is hereditary, we have $O_i \in \mathcal{I}$.

It follows from Lemma 4.1.1 that

$$S_i^1 = \mathsf{Greedy}(V_i) = \mathsf{Greedy}(V_i \cup O_i), \tag{4.5}$$

$$S_i^2 = \mathsf{Greedy}(V_i \setminus S_i^1) = \mathsf{Greedy}((V_i \setminus S_i^1) \cup O_i). \tag{4.6}$$

By combining the equations above with the greedy property (GP), we obtain

$$\begin{aligned} f(S_i^1) &\overset{(4.5)}{=} f(\mathsf{Greedy}(V_i \cup O_i)) \\ &\overset{(\mathrm{GP})}{\geq} \gamma \cdot f(\mathsf{Greedy}(V_i \cup O_i) \cup O_i) \\ &\overset{(4.5)}{=} \gamma \cdot f(S_i^1 \cup O_i), \end{aligned} \tag{4.7}$$

$$\begin{aligned} f(S_i^2) &\overset{(4.6)}{=} f(\mathsf{Greedy}((V_i \setminus S_i^1) \cup O_i)) \\ &\overset{(\mathrm{GP})}{\geq} \gamma \cdot f(\mathsf{Greedy}((V_i \setminus S_i^1) \cup O_i) \cup O_i) \\ &\overset{(4.6)}{=} \gamma \cdot f(S_i^2 \cup O_i). \end{aligned} \tag{4.8}$$

Now we observe that

$$\begin{aligned} f(S_i^1 \cup O_i) + f(S_i^2 \cup O_i) &\geq f((S_i^1 \cup O_i) \cap (S_i^2 \cup O_i)) + f(S_i^1 \cup S_i^2 \cup O_i) && (f \text{ is submodular}) \\ &= f(O_i) + f(S_i^1 \cup S_i^2 \cup O_i) && (S_i^1 \cap S_i^2 = \emptyset) \\ &\geq f(O_i). && (f \text{ is non-negative}) \end{aligned} \tag{4.9}$$

By combining (4.7), (4.8), and (4.9), we obtain

$$f(S_i^1) + f(S_i^2) \geq \gamma \cdot f(O_i). \tag{4.10}$$

Since the distribution of V_i is the same as $\mathcal{V}(1/m)$, for each element $e \in \mathrm{OPT}$, we have

$$\begin{aligned} \Pr[e \in O_i] &= 1 - \Pr[e \notin O_i] = 1 - p_e, \\ \mathbf{E}[\mathbf{1}_{O_i}] &= \mathbf{1}_{\mathrm{OPT}} - \mathbf{p}. \end{aligned} \tag{4.11}$$

By combining (4.10), (4.11), and Lemma 2.3.1, we obtain

$$\begin{aligned}\mathbf{E}[f(S_i^1) + f(S_i^2)] &\geq \gamma \cdot \mathbf{E}[f(O_i)] && \text{(By (4.10))}\\ &\geq \gamma \cdot f^-(\mathbf{1}_{\mathrm{OPT}} - \mathbf{p}). && \text{(By (4.11) and Lemma 2.3.1)}\end{aligned}$$

□

Lemma 4.3.2. $\mathbf{E}[f(\mathsf{Alg}(S))] \geq \beta \cdot f^-(\mathbf{p})$.

Proof. Recall that $S_i^1 = \mathsf{Greedy}(V_i)$, $S_i^2 = \mathsf{Greedy}(V_i \setminus S_i^1)$, and $S = \bigcup_i (S_i^1 \cup S_i^2)$. Since $\mathrm{OPT} \in \mathcal{I}$ and $\mathcal{I}$ is hereditary, $S \cap \mathrm{OPT} \in \mathcal{I}$. Since Alg is a β-approximation, we have

$$f(\mathsf{Alg}(S)) \geq \beta \cdot f(S \cap \mathrm{OPT}). \tag{4.12}$$

Consider an element $e \in \mathrm{OPT}$. For each machine i, we have

$$\begin{aligned}&\Pr[e \in S \mid e \text{ is assigned to machine } i]\\ &= \Pr[e \in \mathsf{Greedy}(V_i) \textbf{ or } e \in \mathsf{Greedy}(V_i \setminus \mathsf{Greedy}(V_i)) \mid e \in V_i]\\ &= \Pr_{A \sim \mathcal{V}(1/m)}[e \in \mathsf{Greedy}(A) \textbf{ or } e \in \mathsf{Greedy}(A \setminus \mathsf{Greedy}(A)) \mid e \in A]\\ &= \Pr_{B \sim \mathcal{V}(1/m)}[e \in \mathsf{Greedy}(B \cup \{e\}) \textbf{ or } e \in \mathsf{Greedy}((B \cup \{e\}) \setminus \mathsf{Greedy}(B \cup \{e\}))]\\ &= p_e.\end{aligned}$$

The first equality above follows from the fact that e is included in S iff e is included in either S_i^1 or S_i^2. The second equality follows from the fact that the distribution of V_i is the same as $\mathcal{V}(1/m)$. The third equality follows from the fact that the distribution of $A \sim \mathcal{V}(1/m)$ conditioned on $e \in A$ is identical to the distribution of $B \cup \{e\}$ where $B \sim \mathcal{V}(1/m)$. Therefore

$$\begin{aligned}\Pr[e \in S \cap \mathrm{OPT}] &= p_e,\\ \mathbf{E}[\mathbf{1}_{S \cap \mathrm{OPT}}] &= \mathbf{p}.\end{aligned} \tag{4.13}$$

By combining (4.12), (4.13), and Lemma 2.3.1, we obtain

$$\mathbf{E}[f(\mathsf{Alg}(S))] \geq \beta \cdot \mathbf{E}[f(S \cap \mathrm{OPT})] \geq \beta \cdot f^-(\mathbf{p}).$$

□

We can now combine Lemmas 4.3.2 and 4.3.1 to obtain our main result for non-monotone submodular maximization.

Theorem 4.3.3. *Let $f : 2^V \to \mathbb{R}_+$ be a submodular function, let $\mathcal{I} \subseteq 2^V$ be a hereditary set system, and let* Alg *is a β-approximation algorithm for* $\max_{S \in \mathcal{I}} f(S)$. *Suppose* Greedy *satisfies* (GP) *with factor γ. The algorithm* NMRandGreeDi *is (in expectation) a $\frac{\beta\gamma}{2\beta+\gamma}$-approximation algorithm for the same problem.*

Proof. Let $S_i^1 = \mathsf{Greedy}(V_i)$, $S_i^2 = \mathsf{Greedy}(V_i \setminus S_i^1)$, and $S = \bigcup_i (S_i^1 \cup S_i^2)$ be the set of elements on the last machine, and $T = \mathsf{Alg}(S)$ be the solution produced on the last machine. Then, the output D produced by RandGreeDi satisfies $f(D) \geq \max_i \max\{f(S_i^1), f(S_i^2)\}$ and $f(D) \geq f(T)$. Thus, from Lemmas 4.3.1 and 4.3.2 we have:

$$\mathbf{E}[f(D)] \geq \frac{\gamma}{2} \cdot f^-(\mathbf{1}_{\mathrm{OPT}} - \mathbf{p}), \tag{4.14}$$

$$\mathbf{E}[f(D)] \geq \beta \cdot f^-(\mathbf{p}). \tag{4.15}$$

By combining (4.14) and (4.15), we obtain

$$\begin{aligned}(2\beta + \gamma)\,\mathbf{E}[f(D)] &\geq \beta\gamma[f^-(\mathbf{p}) + f^-(\mathbf{1}_{\mathrm{OPT}} - \mathbf{p})] \\ &\geq \beta\gamma \cdot f^-(\mathbf{1}_{\mathrm{OPT}}) \\ &= \beta\gamma \cdot f(\mathrm{OPT}).\end{aligned}$$

In the second inequality, we have used the fact that f^- is convex and $f^-(c \cdot \mathbf{x}) \geq c f^-(\mathbf{x})$ for any constant $c \in [0, 1]$. □

Finally, we remark that one can use the following approach on the last machine, and use a lemma from Gupta et al. [2010]. As in the first round, we run Greedy twice to obtain two solutions $T_1 = \mathsf{Greedy}(S)$ and $T_2 = \mathsf{Greedy}(S \setminus T_1)$. Additionally, we select a subset $T_3 \subseteq T_1$ using an *unconstrained* submodular maximization algorithm on T_1, such as the Double Greedy algorithm of [Buchbinder et al., 2012], which is a $\frac{1}{2}$-approximation. The final solution T is the best solution among T_1, T_2, T_3. If Greedy satisfies property GP, then it follows from the analysis of [Gupta et al., 2010] that the resulting solution T satisfies $f(T) \geq \frac{\gamma}{2(1+\gamma)} \cdot f(\mathrm{OPT})$ (shown in the following). This gives us the following corollary of Theorem 4.3.3:

Corollary 4.3.4. *Let $f : 2^V \to \mathbb{R}_+$ be a submodular function, and let $\mathcal{I} \subseteq 2^V$ be a hereditary set system. Suppose* Greedy *satisfies* (GP) *with factor γ. The resulting algorithm described above (which uses* Greedy *twice and* DoubleGreedy*) is (in expectation) a $\frac{\gamma}{4+2\gamma}$-approximation for $\max_{S \in \mathcal{I}} f(S)$.*

Proof. By (GP) and the approximation guarantee of the Double Greedy algorithm, we have:

$$f(T) \geq f(T_1) \geq \gamma \cdot f(T_1 \cup \mathrm{OPT}) \tag{4.16}$$

$$f(T) \geq f(T_2) \geq \gamma \cdot f(T_2 \cup (\mathrm{OPT} \setminus T_1)) \tag{4.17}$$

$$f(T) \geq f(T_3) \geq \frac{1}{2} f(T_1 \cap \mathrm{OPT}). \tag{4.18}$$

Additionally, from [Gupta et al., 2010, Lemma 2], we have:

$$f(T_1 \cup \mathrm{OPT}) + f(T_2 \cup (\mathrm{OPT} \setminus T_1)) + f(T_1 \cap \mathrm{OPT}) \geq f(\mathrm{OPT})$$

By combining the inequalities above, we obtain:

$$(1+\gamma) f(T) \geq \frac{\gamma}{2} \left(f(T_1 \cup \mathrm{OPT}) + f(T_2 \cup (\mathrm{OPT} \setminus T_1)) + f(T_1 \cap \mathrm{OPT}) \right) \geq \frac{\gamma}{2} f(\mathrm{OPT})$$

and hence $f(T) \geq \frac{\gamma}{2(1+\gamma)} \cdot f(\mathrm{OPT})$ as claimed. Setting $\beta = \frac{\gamma}{2(\gamma+1)}$ in Theorem 4.3.3, we obtain an approximation ratio of $\frac{\gamma}{4+2\gamma}$. □

4.4 An improved distributed greedy algorithm for non-monotone submodular maximization

We give here an improved analysis of two-round algorithm for non-monotone submodular maximization subject to a any hereditary constraint. The algorithm is similar to that in Chapter 4 for *monotone* maximization; perhaps surprisingly, we show that this approach achieves a good approximation even for non-monotone functions.

4.4.1 Algorithm

The algorithm considered here is the same RandGreeDi algorithm from Section 4.2. We randomly partition the elements onto the m machines, and run Greedy on the elements V_i on machine i to pick a set S_i. We place the sets S_i on a single machine and we run any algorithm Alg on $B \coloneqq \bigcup_i S_i$ to find a solution T. We return the best solution amongst $S_1, \ldots, S_m, T$.

4.4.2 Analysis

We devote the rest of this section to the analysis of the algorithm. The space and communication aspects are as analyzed in Subsection 4.2.2.

Again we make use of the strong greedy property, defined in section 3.1.1. Given an instance $\langle V, \mathcal{I}, f \rangle$ of non-negative submodular maximization, we assume the Greedy algorithm has the following property:

$$\text{For all } S \in \mathcal{I}: \qquad f(\mathsf{Greedy}(V)) \geq \gamma \cdot f(\mathsf{Greedy}(V) \cup S) \tag{GP}$$

Theorem 4.4.1. *Suppose that* Greedy *satisfies the strong greedy property with constant* γ *and let* Alg *be any* β*-approximation for the problem* $\max_{S \in \mathcal{I}} f(S)$*. Then there is a randomized, two-round distributed algorithm that achieves a* $(1 - \frac{1}{m})\frac{\beta\gamma}{\beta+\gamma}$ *approximation in expectation for* $\max_{S \in \mathcal{I}} f(S)$*.*

Proof. As in the analysis in Section 4.2.2, we let, for each element e, probability $p_e = \Pr_{A \sim \mathcal{V}(1/m)}[e \in \mathsf{Greedy}(A \cup \{e\})]$, if $e \in \mathrm{OPT}$, and 0 otherwise. Then, let $\mathbf{p} \in [0,1]^V$ denote the vector whose entries are given by the probabilities p_e.

We first analyze the expected value of the Greedy solutions S_i. Let

$$O_i = \{e \in \mathrm{OPT} \colon e \notin \mathsf{Greedy}(V_i \cup \{e\})\}.$$

By Lemma 4.1.1, $\mathsf{Greedy}(V_i \cup O) = \mathsf{Greedy}(V_i) = S_i$, and by (GP), $f(S_i) \geq \gamma \cdot f(S_i \cup O)$. Considering the first machine,

$$\begin{aligned}
\mathbf{E}[f(S_1)] &\geq \gamma \cdot \mathbf{E}[f(S_1 \cup O)] \\
&= \gamma \cdot \mathbf{E}[f^-(\mathbf{1}_{S_1 \cup O})] \\
&\geq \gamma \cdot f^-(\mathbf{E}[\mathbf{1}_{S_1 \cup O}]) \\
&= \gamma \cdot f^-(\mathbf{E}[\mathbf{1}_{S_1}] + (\mathbf{1}_{\mathrm{OPT}} - \mathbf{p})).
\end{aligned} \tag{4.19}$$

On line three, we have used the fact that f^- is convex and on line four we have used the fact that $\mathbf{E}[\mathbf{1}_{S_1 \cup O}] = \mathbf{E}[\mathbf{1}_{S_1}] + (\mathbf{1}_{\mathrm{OPT}} - \mathbf{p})$ (since for every $e \in \mathrm{OPT}$, the probability of being included in S_1 is p_e).

Now consider the solution T. Since Alg is a β-approximation, we have

$$\begin{aligned}\mathbf{E}[f(T)] &\geq \beta \cdot \mathbf{E}[f(B \cap \mathrm{OPT})] \\ &= \beta \cdot \mathbf{E}[f^-(\mathbf{1}_{B\cap \mathrm{OPT}})] \\ &\geq \beta \cdot f^-(\mathbf{E}[\mathbf{1}_{B\cap \mathrm{OPT}}]) \\ &= \beta \cdot f^-(\mathbf{p}). \end{aligned} \tag{4.20}$$

Similarly to above, we have used the convexity of f^- and the fact that $\mathbf{E}[\mathbf{1}_{B\cap \mathrm{OPT}}] = \mathbf{p}$.

By combining (4.19) and (4.20), and using convexity of f^-, we obtain

$$\frac{1}{\gamma}\,\mathbf{E}[f(S_1)] + \frac{1}{\beta}\,\mathbf{E}[f(T)] \geq f^-(\mathbf{E}[\mathbf{1}_{S_1}] + (\mathbf{1}_{\mathrm{OPT}} - \mathbf{p})) + f^-(\mathbf{p}) \geq 2 \cdot f^-\left(\frac{\mathbf{E}[\mathbf{1}_{S_1}] + \mathbf{1}_{\mathrm{OPT}}}{2}\right).$$

Since $S_1 \subseteq V_1$ and V_1 is a $1/m$ sample of V, we have $\mathbf{E}[\mathbf{1}_{S_1}] \leq \frac{1}{m} \cdot \mathbf{1}_V$. Therefore, using the definition of f^- and the non-negativity of f, we obtain

$$2 \cdot f^-\left(\frac{\mathbf{E}[\mathbf{1}_{S_1}] + \mathbf{1}_{\mathrm{OPT}}}{2}\right) \geq \left(1 - \frac{1}{m}\right) f(\mathrm{OPT}).$$

Thus

$$\max\{\mathbf{E}[f(S_1)], \mathbf{E}[f(T)]\} \geq \left(1 - \frac{1}{m}\right) \frac{\beta\gamma}{\beta + \gamma} \cdot f(\mathrm{OPT}).$$

□

Examples of results. We conclude this section with some examples of approximation guarantees that we can obtain using Theorem 4.4.1. For a matroid constraint, we have $\gamma = 1/2$ and, if we use the Continuous Greedy algorithm for Alg, we have $\beta = 1/e$; thus we obtain a $\left(1 - \frac{1}{m}\right) \frac{1}{2+e}$ approximation. We remark that, for a cardinality constraint, one can strengthen the proof of Theorem 4.4.1 slightly and obtain a $\left(1 - \frac{1}{m}\right) \frac{1}{e} \left(1 - \frac{1}{e}\right)$ approximation; we give the details in the following.

For a p-system constraint, we have $\gamma = 1/(p+1)$. We can use the algorithm of Gupta et al. [2010] for Alg that achieves an approximation $\beta = 3/\left(2p + 4 + \frac{2}{p}\right)$ when combined with the algorithm of Buchbinder et al. [2012] for unconstrained non-monotone submodular maximization. Thus we obtain a $3\left(1 - \frac{1}{m}\right) / \left(5p + 7 + \frac{2}{p}\right)$ approximation.

4.4.3 A better analysis for cardinality constraints

In this section, we show that for a cardinality constraint, we can improve the analysis slightly of the algorithm given previously.

Theorem 4.4.2. *If $\mathcal{I}$ is a cardinality constraint, the two-round algorithm from Subsection 4.4 achieves a $\left(1-\frac{1}{m}\right)\frac{1-\frac{1}{e}}{1+\frac{1}{\beta}\left(1-\frac{1}{e}\right)}$ approximation in expectation for non-monotone submodular maximization, where β is the approximation guarantee of* Alg.

Proof. The analysis is similar to the one in the proof of Theorem 4.4.1, and we describe the main changes in this section. We define O_i as before, and modify the analysis of the solution S_1 as follows. Let S_1^j be the subset of S_1 consisting of the first j elements picked by Greedy, with $S_1^0 = \emptyset$. By the standard analysis of the Greedy algorithm for a cardinality constraint, for each $j \in [k]$, we have

$$f(S_1^j) - f(S_1^{j-1}) \geq \frac{f(S_1^{j-1} \cup O_1) - f(S_1^{j-1})}{k},$$

and therefore

$$f(S_1) \geq \sum_{j=0}^{k-1} \frac{1}{k}\left(1-\frac{1}{k}\right)^{k-1-j} f(S_1^j \cup O_1).$$

Now, using $\mathbf{E}[\mathbf{1}_{S_1^j \cup O_1}] = \mathbf{E}[\mathbf{1}_{S_1^j}] + \mathbf{E}[\mathbf{1}_{O_1}] = \mathbf{E}[\mathbf{1}_{S_1^j}] + \mathbf{1}_{\mathrm{OPT}} - \mathbf{p}$, and Lemma 2.3.1, we obtain:

$$\mathbf{E}[f(S_1^j \cup O_1)] \geq f^-(\mathbf{E}[\mathbf{1}_{S_1^j}] + \mathbf{1}_{\mathrm{OPT}} - \mathbf{p}).$$

Therefore

$$\mathbf{E}[f(S_1)] \geq \sum_{j=0}^{k-1} \frac{1}{k}\left(1-\frac{1}{k}\right)^{k-1-j} f^-(\mathbf{E}[\mathbf{1}_{S_1^j}] + \mathbf{1}_{\mathrm{OPT}} - \mathbf{p}). \tag{4.21}$$

We analyze the expected value of the solution T as before, obtaining (4.20) from page 39. By combining (4.21) and (4.20), we get

$$\begin{aligned}
&\mathbf{E}[f(S_1)] + \frac{1}{\beta}\left(1-\left(1-\frac{1}{k}\right)^k\right)\mathbf{E}[f(T)] \\
&\quad\geq \sum_{j=0}^{k-1} \frac{1}{k}\left(1-\frac{1}{k}\right)^{k-1-j}\left(f^-(\mathbf{E}[\mathbf{1}_{S_1^j}] + \mathbf{1}_{\mathrm{OPT}} - \mathbf{p}) + f^-(\mathbf{p})\right) \\
&\quad\geq \sum_{j=0}^{k-1} \frac{1}{k}\left(1-\frac{1}{k}\right)^{k-1-j} 2 \cdot f^-\left(\frac{\mathbf{E}[\mathbf{1}_{S_1^j}] + \mathbf{1}_{\mathrm{OPT}}}{2}\right),
\end{aligned}$$

where the last inequality follows from the convexity of f^-. Since $S_1^j \subseteq V_1$ and V_1 is a $1/m$ sample of V, we have $\mathbf{E}[\mathbf{1}_{S_1^j}] \leq \frac{1}{m} \cdot \mathbf{1}_V$. Therefore, using the definition of f^- and

Algorithm 7: Descending Thresholds Greedy algorithm (DThreshGreedy).

```
Input: N ⊆ V
S ← ∅, d ← max_{e∈N} f({e})
for w = d; w ≥ (ε/n)d; w ← w(1 − ε) do
    foreach e ∈ N do
        if S ∪ {e} ∈ I
         and f(S ∪ {e}) − f(S) ≥ w then
            S ← S ∪ {e}
return S
```

Figure 4.1: The Descending Thresholds Greedy algorithm of [Kumar et al., 2013; Badanidiyuru and Vondrák, 2014].

the non-negativity of f, we obtain

$$2 \cdot f^{-}\left(\frac{\mathbf{E}[\mathbf{1}_{S_1^j}] + \mathbf{1}_{\mathrm{OPT}}}{2}\right) \geq f^{-}\left(\mathbf{E}[\mathbf{1}_{S_1^j}] + \mathbf{1}_{\mathrm{OPT}}\right) \geq \left(1 - \frac{1}{m}\right) f(\mathrm{OPT}),$$

where the first inequality follows from properties of f^{-}, and the second from the fact that $(\mathbf{E}[\mathbf{1}_{S_1^j}] + \mathbf{1}_{\mathrm{OPT}})_e \leq 1/m$ for $e \in V \backslash \mathrm{OPT}$. Thus,

$$\mathbf{E}[f(S_1)] + \frac{1}{\beta}\left(1 - \left(1 - \frac{1}{k}\right)^k\right)\mathbf{E}[f(T)] \geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right)\left(1 - \frac{1}{m}\right) f(\mathrm{OPT}).$$

It follows that

$$\begin{aligned}\max\left\{\mathbf{E}[f(S_1)], \mathbf{E}[f(T)]\right\} &\geq \left(1 - \frac{1}{m}\right) \frac{\left(1 - \left(1 - \frac{1}{k}\right)^k\right)}{1 + \frac{1}{\beta}\left(1 - \left(1 - \frac{1}{k}\right)^k\right)} f(\mathrm{OPT}) \\ &\geq \left(1 - \frac{1}{m}\right) \frac{1 - \frac{1}{e}}{1 + \frac{1}{\beta}\left(1 - \frac{1}{e}\right)} f(\mathrm{OPT}).\end{aligned}$$

□

4.5 A fast sequential algorithm for matroid constraints

We now show how our approach can be used to obtain a fast *sequential* algorithm for non-monotone maximization subject to a matroid constraint. Notice that the analysis given in Theorem 4.4.1 only relies on the following two properties of the Greedy algorithm: it satisfies (GP) and Lemma 4.1.1. Thus we can replace the Greedy algorithm by any

Algorithm 8: Sequential algorithm for non-monotone submodular maximization subject to a matroid constraint

Input: instance $\langle V, \mathcal{I}, f \rangle$
Let $m = 1/\epsilon$ and initialize $V_1, \ldots, V_m$ to the empty set
for $e \in V$ **do**
  Assign e to V_i uniformly at random
Run $\mathsf{DThreshGreedy}(V_i)$ on each sample V_i to obtain S_i
Let $B = \bigcup_i S_i$
Run $\mathsf{Alg}(B)$ to obtain B'
Let $A = \arg\max_i \{f(S_i)\}$
return $\arg\max\{f(A), f(B')\}$

algorithm satisfying these two properties. In particular, the Descending Thresholds Greedy (shown in Figure 4.1 as DThreshGreedy) of [Kumar et al., 2013; Badanidiyuru and Vondrák, 2014] satisfies these conditions with $\gamma = 1/2 - \epsilon$ (follows from analysis in [Badanidiyuru and Vondrák, 2014]).

Our algorithm proceeds as follows. We randomly partition the elements into $m \coloneqq 1/\epsilon$ samples $V_1, V_2, \ldots, V_m$. On each sample, we run the Descending Thresholds Greedy algorithm on V_i to obtain a solution S_i. Let $A \coloneqq \text{argmax}_{i \in [m]} f(S_i)$ and $B \coloneqq \bigcup_i S_i$. Then, $|B| \leq k/\epsilon$, where k is the rank of the matroid. We run any β-approximation algorithm Alg on B to find a solution B', and we return the better of A and B'. We obtain the following result.

Theorem 4.5.1. *There is a sequential, randomized* $(\frac{1}{2+\epsilon} - \epsilon)$*-approximation algorithm for the problem* $\max_{S \in \mathcal{I}} f(S)$*, where* $\mathcal{I}$ *is any matroid constraint, running in time* $O(\frac{n}{\epsilon} \log n) + \text{poly}(\frac{k}{\epsilon})$.

Proof. The running time of the Descending Thresholds Greedy algorithm on a ground set of size s is $O(\frac{s}{\epsilon} \log(\frac{s}{\epsilon}))$. Each random sample has size $O(\epsilon n)$ with high probability[3], and thus the total time needed to construct B is $O(\frac{n}{\epsilon} \log n)$ with high probability. It follows from the analysis in Theorem 4.4.1 that the best of the two solutions A and a β-approximation to $\max_{S \subseteq B : S \in \mathcal{I}} f(S)$ is a $\frac{1}{2+\frac{1}{\beta}} - \epsilon$ approximation. We can then use any $1/e$-approximation (such as the Continuous Greedy)algorithm as Alg. □

[3]Each sample has expected size ϵn. Using Chernoff bounds, we obtain that the probability the size of a sample is at most $2\epsilon n$ is greater than $1 - (e/4)^{\epsilon n}$

Chapter 5

Multi-round distributed algorithms for constrained submodular maximization

Given the constant approximation factors obtained by the RandGreeDi algorithm (Chapter 4) one cannot help but wonder whether it would be possible to push the techniques further and obtain a generic method to carry over the algorithms in the sequential setting to the parallel world, taking advantage of the improvements made for these over the last few decades.

In this chapter, we extend our approach from Chapter 4. Our main contribution is a *generic parallel algorithm* that allows us to parallelize a broad class of sequential algorithm with almost no loss in performance. The crux of our approach is a *common abstraction* that allows us to capture and parallelize both the standard and continuous greedy algorithms, and it provides a novel unifying perspective for these algorithmic paradigms. Our framework leads to the first distributed algorithms that nearly match the state of the art approximation guarantees for the sequential setting in only a constant number of rounds.

In contrast with the previous framework by Kumar et al. [2013] which is based on repeatedly eliminating bad elements, our framework is more in line with the greedy approach of identifying good elements. The algorithm maintains a pool of good elements that is grown over several rounds. In each round, the elements are partitioned randomly into groups. Each group selects the best among its elements and the good pool using the sequential algorithm. Finally, the best elements from all groups are added to the good pool. The best solution among the ones found in the execution of the algorithm is returned at the end. The previous works based on 2 rounds of MapReduce such as the one presented in Chapter 4 can be viewed as a single phase of our algorithm. The first phase can already identify a constant fraction of the weight of the solution, thus

obtaining a constant factor approximation. However, it is not clear how to obtain the best approximation factor from such an approach. Our main insight is that, with a right measure of progress, we can grow the solution iteratively and obtain solutions that are arbitrarily close to those of sequential algorithms. We show that after a few rounds, the pool of good elements already contains a good solution with constant probability.

In the following, we summarize our main contributions in this chapter.

A parallel greedy algorithm. We obtain the following general result by parallelizing the standard greedy algorithm:

Let $f : 2^V \to \mathbb{R}_+$ be a submodular function, and $\mathcal{I} \subseteq 2^V$ be a hereditary set system For any $\epsilon > 0$ there is a randomized distributed $O(1/\epsilon)$-round algorithm that can be implemented in the MapReduce framework. The algorithm is an $(\alpha - O(\epsilon))$-approximation with constant probability for the problem $\max_{S \in \mathcal{I}} f(S)$, where α is the approximation ratio of the standard, sequential greedy algorithm for the same problem.

Our constant number of rounds is a significant improvement over the sample and prune technique of Kumar et al. [2013], which requires a number of rounds depending logarithmically on the *value* of the single best element. Remarkably, even for the especially simple case of a cardinality constraint, no previous work could get close to the approximation ratio of the simple sequential greedy algorithm in a constant number of rounds. Our framework nearly matches the approximation ratio of greedy in all situations in a constant number of rounds and immediately resolves this problem.

A parallel continuous greedy algorithm. We obtain new distributed approximation results for maximization over matroids, by using a heavily discretized variant of the measured continuous greedy algorithm, obtaining approximation guarantees nearly matching those attained by the continuous greedy in the sequential setting.

Let $f : 2^V \to \mathbb{R}_+$ be a submodular function, and $\mathcal{I} \subseteq 2^V$ be a matroid. For any $\epsilon > 0$ there is a randomized distributed $O(1/\epsilon)$-round algorithm that can be implemented in the MapReduce framework. The algorithm is an $(\alpha - O(\epsilon))$-approximation with constant probability for the problem $\max_{S \in \mathcal{I}} f(S)$, where α is $(1 - 1/e)$ for monotone f and $1/e$ for general f.

Simple two-round algorithms. Lastly, we show how the techniques developed in the previous sections can be used to obtain a very simple two-round distributed algorithm for monotone maximization subject to a cardinality constraint.

There is a randomized, two-round, distributed algorithm achieving a $\frac{1}{2} - \epsilon$ ap-

proximation (in expectation) for $\max_{S:|S|\leq k} f(S)$*, where* f *is a monotone function.*

5.1 Generic parallel algorithm for submodular maximization

In this section, we give a generic approach for parallelizing *any sequential algorithm* Alg for the problem $\max_{S\subseteq V:\, S\in\mathcal{I}} f(S)$, where $f : 2^V \to \mathbb{R}_+$ is a submodular function and $\mathcal{I} \subseteq 2^V$ is a hereditary constraint.

As a starting point, we need a common abstract description of existing sequential algorithms. Towards that end, we turn to the standard Greedy and Continuous Greedy algorithms for inspiration. The Greedy algorithm directly constructs a solution, whereas the Continuous Greedy algorithm first constructs a fractional solution $\mathbf{x}$ which is then rounded to get an integral solution. In the common abstraction, we will need both the integral solution *and* the support of the fractional solution $\mathbf{x}$. To account for this, we will have the algorithm Alg return a pair of sets, $(\mathsf{AlgSol}(V), \mathsf{AlgRel}(V))$, where $\mathsf{AlgSol}(V) \in I$ is a feasible solution for the problem and $\mathsf{AlgRel}(V)$ is a set providing additional information. When using the standard Greedy algorithm for Alg, $\mathsf{AlgSol}(V)$ and $\mathsf{AlgRel}(V)$ will both be equal to the Greedy solution. When using the Continuous Greedy algorithm for Alg, $\mathsf{AlgSol}(V)$ will be the integral solution and $\mathsf{AlgRel}(V)$ will be the support of the fractional solution constructed by the Continuous Greedy algorithm.

More importantly, we will need an abstraction that captures the greedy behavior of these algorithms. We encapsulate the crucial properties of greedy-like algorithms in the following definition. We believe that this framework is one of the most valuable and insightful contributions of this work, and it provides a general abstraction for a broader class of algorithms.

We assume that the algorithm Alg satisfies the following properties.

1. (α-Approximation) For every input $N \subseteq V$, $\mathsf{AlgSol}(N)$ is an α-approximate solution to $\max_{S\subseteq N:\, S\in\mathcal{I}} f(S)$.

2. (Consistency) Let A and B be two disjoint subsets of V. Suppose that, for each element $e \in B$, we have $\mathsf{AlgRel}(A \cup \{e\}) = \mathsf{AlgRel}(A)$. Then $\mathsf{AlgSol}(A \cup B) = \mathsf{AlgSol}(A)$.

Armed with this definition, we can now describe our approach for parallelizing an abstract sequential algorithm Alg with almost no loss in the approximation guarantee.

Parallel algorithm $\mathsf{ParallelAlg}$ **based on** Alg**.** As before, let α be the approximation guarantee of the sequential algorithm Alg. Let $s \coloneqq \max_{N\subseteq V} |\mathsf{AlgSol}(N) \cup \mathsf{AlgRel}(N)|$ be

Algorithm 9: The distributed algorithm ParallelAlg

Input: instance $\langle V, \mathcal{I}, f \rangle$, number g of groups, number m of machines, desired accuracy ϵ

$S_{\text{best}} \leftarrow \emptyset$, $C_0 \leftarrow \emptyset$
for *run* $r \in \Theta(1/\epsilon)$ **do**
  for *each group of machines* **do**
    for $e \in V$ **do**
      Assign e to a machine $i \in [m]$ chosen uniformly at random
  Let $X_{i,r}$ be the elements assigned to machine i in run r ($i \in [gm]$)
  Run $\mathsf{Greedy}(X_{i,r} \cup C_{r-1})$ on each machine i to obtain $(\mathsf{AlgSol}(X_{i,r} \cup C_{r-1}), \mathsf{AlgRel}(X_{i,r} \cup C_{r-1}))$
  Update $S_{\text{best}} \leftarrow \arg\max_i \{f(S_{\text{best}}), f(\mathsf{AlgSol}(X_{i,r} \cup C_{r-1}))\}$
  Update $C_r \leftarrow C_{r-1} \bigcup_i \mathsf{AlgRel}(X_{i,r} \cup C_{r-1})$;
return S_{best}

the maximum size of the sets returned by Alg. Let $\epsilon > 0$ be the desired accuracy, i.e., we will aim that ParallelAlg achieves an $(\alpha - \epsilon)$ approximation.

The algorithm uses $g \coloneqq \Theta(1/(\alpha\epsilon))$ groups of machines with m machines in each group (and thus the total number of machines is gm). The number m of machines can be chosen arbitrarily and it will determine the amount of space needed on each machine, since the dataset is divided roughly equally among each of the m machines in each group. An optimal setting, in terms of space and communication, is $gm \coloneqq O(\sqrt{n/s})$.

The algorithms performs $\Theta(1/\epsilon)$ *runs*. Throughout the process, we maintain two quantities: an *incumbent* solution S_{best}, which is the best solution produced on any single machine so far in the process, and a *pool* of elements $C \subseteq V$ (we assume that the incumbent solution is stored on one designated machine).

Each run of the algorithm proceeds as follows. Amongst each group of m machines, we partition V uniformly at random; each element e chooses an index $i \in [m]$ uniformly and independently at random and is assigned to the ith machine in the group. We do this separately for each group of machines, i.e., each element appears on exactly one machine in each group. For an individual machine $i \in [gm]$, let $X_{i,r}$ denote the set of elements that are assigned to i in run r by this procedure. Additionally, we place on each machine the same pool of elements C_{r-1}, constructed at the end of run $r-1$.

Once the elements have been distributed as described above, on each machine i, we run the algorithm Alg on the input $X_{i,r} \cup C_{r-1}$ on the machine to obtain $(\mathsf{AlgSol}(X_{i,r} \cup C_{r-1}), \mathsf{AlgRel}(X_{i,r} \cup C_{r-1}))$. We update the incumbent solution S_{best} to be the better of the current solution S_{best} and the solutions $\mathsf{AlgSol}(X_{i,r} \cup C_{r-1})$ constructed on each

of the machines; this is achieved by having each machine send $\mathsf{AlgSol}(X_{i,r} \cup C_{r-1})$ to some designated machine maintaining S_{best}, and this machine will update S_{best} in the next round. We update the pool by setting $C_r \coloneqq C_{r-1} \bigcup_i \mathsf{AlgRel}(X_{i,r} \cup C_{r-1})$; this is achieved by having each machine send $\mathsf{AlgRel}(X_{i,r} \cup C_{r-1})$ to every other machine, and thus ensuring that the pool C_r is available on each machine during the next round.

At the end of the $\Theta(1/\epsilon)$ runs, the algorithm returns the incumbent solution S_{best}. This completes the description of our algorithm.

Avoiding duplicating the dataset. The algorithm above partitions the dataset over $\Theta(1/\epsilon)$ groups of machines and thus it duplicates the dataset $\Theta(1/\epsilon)$ times (this problem also applies to previous work [Mirrokni and Zadimoghaddam, 2015]). This is done in order to achieve the best theoretical guarantee on the number of runs, but in practice it is undesirable to duplicate the data. Instead, we can use a single group of m machines and perform the computation of a single run sequentially over $\Theta(1/\epsilon)$ sub-run, where each sub-run performs the computation of one of the group of machines. This will lead to an algorithm that performs $\Theta(1/\epsilon^2)$ runs using m machines and it does not duplicate the dataset.

The analysis. We devote the rest of this section to the analysis of the algorithm ParallelAlg. We start by noting that, if we choose g and m so that $gm = O(\sqrt{n/s})$, the algorithm uses the following resources and thus it satisfies the requirements of the model in Section 2.4.

Lemma 5.1.1. ParallelAlg *can be implemented in the parallel model in Section 2.4 using the following resources.*

- *The number of rounds is* $O(1/\epsilon)$.
- *The number of machines is* $O(\sqrt{n/s})$.
- *The amount of space used on each machine is* $O(\sqrt{ns}/(\epsilon\alpha))$ *with high probability.*
- *In each round, the total amount of communication from a machine to all other machines is* $O(\sqrt{ns}/(\epsilon\alpha))$ *with high probability. The total amount of communication over all machines in a given round is* $O(n/(\epsilon\alpha))$.

Proof. We will choose $gm \coloneqq \sqrt{n/s}$ as our number of machines. Using this choice, we can provide the guarantees stated in the lemma.

Note that we can combine the update step of the incumbent solution and the pool of a given run with the next run's distribution of elements into a single round of communication. Specifically, each machine computes a new random assignment for each

element of its sample $X_{i,r}$, assigns all of its new pool elements to *all* machines, and sends its solution to the designated machine. Thus each run corresponds to a round of communication. In each round, a machine communicates its sample $X_{i,r}$, which has size $O(n/m) = O(\sqrt{ns}/(\epsilon\alpha))$ with high probability[1], and the sets $\mathsf{AlgSol}(X_{i,r} \cup C_{r-1})$ and $\mathsf{AlgRel}(X_{i,r} \cup C_{r-1})$ that have size $O(s)$ to all other machines. Thus the total amount that a machine communicates is $O(\sqrt{ns}/(\epsilon\alpha) + s \cdot gm) = O(\sqrt{ns}/(\epsilon\alpha))$ with high probability, and the total amount that all machines communicate is $O(n + n/m \cdot gm) = O(n/(\epsilon\alpha))$.

In every round, the space used on a given machine is the size of its sample $X_{i,r}$, which is $O(n/m) = O(\sqrt{ns}/(\epsilon\alpha))$ with high probability; the size of the incumbent solution, which is $O(s)$; and the size of the pool, which is $O(gm \cdot s/\epsilon) = O(\sqrt{ns}/\epsilon)$. Therefore the total amount of space used on each machine is $O(\sqrt{ns}/(\epsilon\alpha))$ with high probability. □

Thus it remains to analyze the quality of the solution constructed by the algorithm. In the remainder of this section, we show that, if Alg satisfies the α-approximation and consistency properties defined above, the parallel algorithm $\mathsf{ParallelAlg}$ achieves an $(\alpha - O(\epsilon))$ approximation. For simplicity, in this section we assume that Alg is deterministic; in Section 5.5, we extend our approach to the setting in which Alg is randomized. We start by introducing some notation. As before, let $\mathcal{V}(1/m)$ denote the distribution over random subsets of V where each element is included independently with probability $1/m$. Recall that $X_{i,r} \sim \mathcal{V}(1/m)$ is the random sample placed on machine i at the beginning of run r and C_{r-1} is the pool of elements at the beginning of run r. The following theorem is the crux of our analysis.

Theorem 5.1.2. *Consider a run $r \geq 1$ of the algorithm. Let $\widehat{C}_{r-1} \subseteq V$. Then one of the following must hold:*

(1) $\mathbf{E}_{X_{1,r}}[f(\mathsf{AlgSol}(C_{r-1} \cup X_{1,r})) \mid C_{r-1} = \widehat{C}_{r-1}] \geq (1 - \epsilon)^2\alpha \cdot f(\mathrm{OPT})$, *or*

(2) $\mathbf{E}[f(C_r \cap \mathrm{OPT}) \mid C_{r-1} = \widehat{C}_{r-1}] - f(\widehat{C}_{r-1} \cap \mathrm{OPT}) \geq \frac{\epsilon}{2} \cdot f(\mathrm{OPT})$.

Intuitively, Theorem 5.1.2 shows that, in expectation, if we have not found a good solution on some machine after $O(1/\epsilon)$ runs, then the current pool C, available to every machine, must satisfy $f(C \cap \mathrm{OPT}) = f(\mathrm{OPT})$, and so each machine in the next run will in fact return a solution of quality at least $\alpha f(\mathrm{OPT})$. The following theorem, whose proof we give in Section 5.4, makes this formal.

[1]Each sample has expected size n/m. Using Chernoff bounds, we obtain that the probability the size of a sample is at most $2(n/m)$ is greater than $1 - (e/4)^{(n/m)}$

Theorem 5.1.3. ParallelAlg *achieves an* $(1-\epsilon)^3\alpha$ *approximation with constant probability.*

We devote the rest of this section to the proof of Theorem 5.1.2. Consider a run r of the algorithm. Let $\widehat{C}_{r-1} \subseteq V$. In the following, we condition on the event that $C_{r-1} = \widehat{C}_{r-1}$.

For each element $e \in V$, let $p_r(e) = \Pr_{X\sim\mathcal{V}(1/m)}[e \in \mathsf{AlgRel}(\widehat{C}_{r-1} \cup X \cup \{e\})]$ if $e \in \mathrm{OPT} \setminus \widehat{C}_{r-1}$, and 0 otherwise. As shown in the following lemma, the probability $p_r(e)$ gives us a handle on the probability that e is in the union of the relevant sets.

Lemma 5.1.4. *For each element* $e \in \mathrm{OPT} \setminus \widehat{C}_{r-1}$,

$$\Pr[e \in \cup_{1\le i\le gm}\mathsf{AlgRel}(\widehat{C}_{r-1} \cup X_{i,r})] = 1 - (1-p_r(e))^g,$$

where g *is the number of groups into which the machines are partitioned.*

Proof. For each group G_j, we can show that e is *not* in the union of the relevant sets for that group with probability $1-p_r(e)$. Since different groups have independent partitions, e is not in the union of the relevant sets for all machines with probability $(1-p_r(e))^g$, and the lemma follows. More precisely, for each group G_j, let $\overline{Y}_j$ be the event that $e \notin \bigcup_{i\in G_j} \mathsf{AlgRel}(\widehat{C}_{r-1} \cup X_{i,r})$. Let $G_{j,\ell}$ denote the ℓth machine in G_j. We have

$$\begin{aligned}\Pr\left[\overline{Y}_j\right] = \frac{1}{m}\sum_{\ell=1}^{m}\Pr[\overline{Y}_j \mid e \text{ is on } G_{j,\ell}] &= \frac{1}{m}\sum_{i\in G_j}\Pr_{X_{i,r}}[e \notin \mathsf{AlgRel}(\widehat{C}_{r-1} \cup X_{i,r}) \mid e \in X_{i,r}] \\ &= \frac{1}{m}\sum_{i\in G_j}\Pr_{X\sim\mathcal{V}(1/m)}[e \notin \mathsf{AlgRel}(\widehat{C}_{r-1} \cup X \cup \{e\})] = 1-p_r(e),\end{aligned}$$

where the first equality follows from the fact that e assigned to a machine in G_j chosen independently and uniformly at random, and the third from the fact that the distribution of $X_{\ell,r} \sim \mathcal{V}(1/m)$ conditioned on $e \in X_{\ell,r}$ is identical to the distribution of $X \cup \{e\}$ with $X \sim \mathcal{V}(1/m)$. Since the events $\{Y_j \colon 1 \le j \le g\}$ are mutually independent, $\Pr[\bigwedge_{1\le j\le N}\overline{Y}_j] = \prod_{j=1}^{g}\Pr[\overline{Y}_j] = (1-p_r(e))^g$. □

Returning to the proof of Theorem 5.1.2, we define a partition (P_r, Q_r) of $\mathrm{OPT} \setminus \widehat{C}_{r-1}$ as follows:

$$P_r = \{e \in \mathrm{OPT} \setminus \widehat{C}_{r-1} \colon p_r(e) < \epsilon\} \qquad Q_r = \{e \in \mathrm{OPT} \setminus \widehat{C}_{r-1} \colon p_r(e) \ge \epsilon\}$$

The following subsets of P_r and Q_r are key to our analysis (recall that $X_{i,r}$ is the random sample placed on machine i at the beginning of the run r):

$$P'_r = \left\{e \in P_r \colon e \notin \mathsf{AlgRel}(\widehat{C}_{r-1} \cup X_{1,r} \cup \{e\})\right\} \quad Q'_r = Q_r \cap \left(\cup_{i=1}^{gm} \mathsf{AlgRel}(\widehat{C}_{r-1} \cup X_{i,r})\right).$$

Note that each element $e \in P_r$ is in P'_r with probability $1 - p_r(e) \geq 1 - \epsilon$. Further, by Lemma 5.1.4, each element $e \in Q_r$ is in Q'_r with probability $1-(1-p_r(e))^g \geq 1-\frac{1}{e} \geq \frac{1}{2}$.

It follows from the definition of P'_r and the consistency property of Alg that

$$\mathsf{AlgSol}(\widehat{C}_{r-1} \cup X_{1,r}) = \mathsf{AlgSol}(\widehat{C}_{r-1} \cup X_{1,r} \cup P'_r).$$

Let $\mathrm{OPT}_{r-1} = \widehat{C}_{r-1} \cap \mathrm{OPT}$ be the part of OPT in this iteration's pool. Then, since Alg is an α approximation and $P'_r \cup \mathrm{OPT}_{r-1} \subseteq \mathrm{OPT}$ is a feasible solution, we have

$$f(\mathsf{AlgSol}(\widehat{C}_{r-1} \cup X_{1,r})) \geq \alpha \cdot f(P'_r \cup \mathrm{OPT}_{r-1}).$$

Taking expectations on both sides, we have:

$$\mathbf{E}[f(\mathsf{AlgSol}(\widehat{C}_{r-1} \cup X_{1,r}))] \geq \alpha \cdot \mathbf{E}[f(P'_r \cup \mathrm{OPT}_{r-1})] \geq (1-\epsilon)\alpha \cdot f(P_r \cup \mathrm{OPT}_{r-1}), \quad (5.1)$$

where the final inequality follows from Lemma 2.3.2, since f is monotone when restricted to OPT (Lemma 2.2.1), $P_r \cup \mathrm{OPT}_{r-1} \subseteq \mathrm{OPT}$, and P'_r contains every element of P_r with probability at least $(1-\epsilon)$.

Note that $Q'_r \subseteq (\mathrm{OPT} \cap C_r) \setminus \mathrm{OPT}_{r-1}$. As before, f is monotone when restricted to OPT. Additionally, Q'_r contains every element of Q_r with probability at least $1/2$. Thus,

$$\mathbf{E}[f(C_r \cap \mathrm{OPT}) \mid C_{r-1} = \widehat{C}_{r-1}] \geq \mathbf{E}[f(Q'_r \cup \mathrm{OPT}_{r-1})] \geq \frac{1}{2} \cdot f(Q_r \cup \mathrm{OPT}_{r-1}) + \frac{1}{2} \cdot f(\mathrm{OPT}_{r-1}),$$

where the final inequality follows from Lemma 2.3.2. Subtracting $f(\mathrm{OPT}_{r-1})$ from both sides, we get

$$\mathbf{E}[f(C_r \cap \mathrm{OPT}) \mid C_{r-1} = \widehat{C}_{r-1}] - f(\mathrm{OPT}_{r-1}) \geq \frac{1}{2}\left(f(Q_r \cup \mathrm{OPT}_{r-1}) - f(\mathrm{OPT}_{r-1})\right).$$

Using the condition $C_{r-1} = \widehat{C}_{r-1}$ and the definition $\mathrm{OPT}_{r-1} = \widehat{C}_{r-1} \cap \mathrm{OPT}$ we obtain:

$$\begin{aligned}
&\mathbf{E}[f(C_r \cap \mathrm{OPT}) - f(C_{r-1} \cap \mathrm{OPT}) \mid C_{r-1} = \widehat{C}_{r-1}] \\
&\qquad \geq \frac{1}{2}\left(f(Q_r \cup \mathrm{OPT}_{r-1}) - f(\mathrm{OPT}_{r-1})\right) \\
&\qquad \geq \frac{1}{2}\left(f(P_r \cup Q_r \cup \mathrm{OPT}_{r-1}) - f(P_r \cup \mathrm{OPT}_{r-1})\right) \\
&\qquad = \frac{1}{2}\left(f(\mathrm{OPT}) - f(P_r \cup (\widehat{C}_{r-1} \cap \mathrm{OPT}))\right), \qquad (5.2)
\end{aligned}$$

where the second inequality follows from submodularity.

Now, if $f(P_r \cup (\widehat{C}_{r-1} \cap \mathrm{OPT})) \geq (1-\epsilon) \cdot f(\mathrm{OPT})$ then, by (5.1), we have

$$\mathbf{E}[f(\mathsf{AlgSol}(\widehat{C}_{r-1} \cup X_{1,r}))] \geq (1-\epsilon)\alpha \cdot f(P_r \cup (\widehat{C}_{r-1} \cap \mathrm{OPT})) \geq (1-\epsilon)^2\alpha \cdot f(\mathrm{OPT}),$$

and the first property in the statement of Theorem 5.1.2 holds. Otherwise, $f(\mathrm{OPT}) - f(P_r \cup (\widehat{C}_{r-1} \cap \mathrm{OPT})) \geq \epsilon \cdot f(\mathrm{OPT})$, and we have from (5.2)

$$\begin{aligned}
&\mathbf{E}[f(C_r \cap \mathrm{OPT}) - f(C_{r-1} \cap \mathrm{OPT}) \mid C_{r-1} = \widehat{C}_{r-1}] \\
&\qquad \geq \frac{1}{2}\left(f(\mathrm{OPT}) - f(P_r \cup (\widehat{C}_{r-1} \cap \mathrm{OPT}))\right) \\
&\qquad \geq \frac{\epsilon}{2} \cdot f(\mathrm{OPT}),
\end{aligned}$$

which implies that the second property must hold.

This completes the description of our generic approach. In the following sections, we instantiate the algorithm Alg with the standard Greedy algorithm and a heavily discretized Continuous Greedy algorithm, and obtain our main results stated in the introduction.

5.2 A parallel greedy algorithm

In this section, we combine the generic approach from Section 5.1 with the standard greedy algorithm, and give our results for monotone maximization stated in Theorem 5.2.1.

We let Alg be the standard Greedy algorithm. We let $\mathsf{AlgRel}(N) = \mathsf{AlgSol}(N) = \mathsf{Greedy}(N)$. It was shown in the previous chapter that the Greedy algorithm satisfies the consistency property. We restate the Lemma here for readability.

Lemma 4.1.1. *Let $A \subseteq V$ and $B \subseteq V$ be two disjoint subsets of V. Suppose that, for each element $e \in B$, we have $\mathsf{Greedy}(A \cup \{e\}) = \mathsf{Greedy}(A)$. Then $\mathsf{Greedy}(A \cup B) = \mathsf{Greedy}(A)$.*

Algorithm 10: Discretized Continuous Greedy (DCGreedy).

Input: $N \subseteq V$

1 $\mathbf{x}(0) \leftarrow \mathbf{0}$
2 **for** $t \leftarrow 1$ *to* $1/\epsilon$ **do**
3 $\quad \mathbf{y}(t) \leftarrow \mathsf{GreedyStep}(N, \mathbf{x}(t))$
4 $\quad \mathbf{x}(t) \leftarrow \mathbf{x}(t-1) + \mathbf{y}(t)$
5 $S \leftarrow \mathsf{SwapRounding}(\mathbf{x}(1/\epsilon), \mathcal{I})$
6 Let T be the support of $\mathbf{x}(1/\epsilon)$
7 **return** (S, T)

Algorithm 11: Greedy Update Step (GreedyStep).

Input: $N \subseteq V$, $\mathbf{x} \in [0,1]^N$

1 $W \leftarrow \emptyset$, $\mathbf{y} \leftarrow \mathbf{0}$
2 **repeat**
3 $\quad D \leftarrow \{e \in N \setminus W : W \cup \{e\} \in \mathcal{I}\}$
4 $\quad$ **foreach** $e \in D$ **do**
5 $\quad\quad w_e \leftarrow \mathbf{E}[f(R(\mathbf{x}+\mathbf{y}) \cup \{e\}) - f(R(\mathbf{x}+\mathbf{y}))]$
6 $\quad$ Let $e^* = \arg\max_{e \in D} w_e$
7 $\quad$ **if** $D = \emptyset$ *or* $w_{e^*} < 0$ **then return** $\mathbf{y}$
8 $\quad$ **else** $y_{e^*} \leftarrow y_{e^*} + \epsilon(1 - x_{e^*})$
9 $\quad W \leftarrow W \cup \{e^*\}$

Figure 5.1: The discretized continuous greedy algorithm. On line 5 of Algorithm 11, for a vector $\mathbf{z} \in [0,1]^N$, we use $R(\mathbf{z})$ to denote a random subset of N that contains each element e independently with probability z_e. The weights on line 5 cannot be computed exactly in polynomial time, but they can be efficiently approximated using random samples.

The Lemma allows us to immediately apply the result from Section 5.1 and obtain the following.

Theorem 5.2.1. *Let $f : 2^V \to \mathbb{R}_+$ be a submodular function, and $\mathcal{I} \subseteq 2^V$ be a hereditary set system. For any $\epsilon > 0$ there is a randomized distributed $O(1/\epsilon)$-round algorithm that can be implemented in the MapReduce model (described in Section 2.4). The algorithm is an $(\alpha - O(\epsilon))$-approximation with constant probability for the problem $\max_{S \in \mathcal{I}} f(S)$, where α is the approximation ratio of the standard, sequential greedy algorithm for the same problem.*

5.3 A parallel continuous greedy algorithm

For monotone maximization subject to a matroid constraint, Theorem 5.2.1 guarantees only a $(1/2 - \epsilon)$ approximation, due to the limitations of the standard greedy algorithm. We obtain a nearly optimal $(1 - 1/e - \epsilon)$ approximation by instantiating the framework in Section 5.1 with the DCGreedy algorithm shown in Algorithm 10.

The DCGreedy algorithm is a heavily discretized version of the Continuous Greedy approach of Feldman et al. [2011], and it first constructs an approximate fractional solution to the problem $\max_{\mathbf{x} \in P(\mathcal{I})} F(\mathbf{x})$ of maximizing the multilinear extension F of f subject to the constraint that $\mathbf{x}$ is in the matroid polytope $P(\mathcal{I})$, and then rounds

the fractional solution without loss using PipageRounding or SwapRounding [Ageev and Sviridenko, 2004; Chekuri et al., 2010].

The algorithm finds, at each timestep t, an update vector $\mathbf{y}(t)$ via the GreedyStep routine. GreedyStep, in turn, works like one step of the Continuous Greedy algorithm. However, instead of finding any maximizing update vector, it finds an independent set W, and increases all $e \in W$ by $\epsilon(1 - x_e)$.

In this section, we combine the generic approach from Section 5.1 with the DCGreedy algorithm. We use DCGreedy as Alg; the relevant set AlgRel(N) is the set of elements in the support of the fractional solution $\mathbf{x}(1/\epsilon)$, and AlgSol(N) is the integral solution obtained by rounding $\mathbf{x}(1/\epsilon)$.

Note that it is necessary to ensure that the fractional solution has small support so that the size of AlgRel(N) is small. We achieve this by heavily discretizing the continuous greedy algorithm, thereby limiting the number of support updates performed in lines 3 and 4 of DCGreedy. Unfortunately, performing this discretization naively introduces an error in the approximation that is too large. Thus, we make use of a key idea from [Badanidiyuru and Vondrák, 2014], which can be applied in the case of a matroid constraint. This allows us to show the following lemma whose proof is deferred to Section 5.6.

Lemma 5.3.1. *The* DCGreedy *algorithm achieves an* $(1-1/e-O(\epsilon))$ *approximation for monotone functions and an* $(1/e - O(\epsilon))$ *approximation for non-monotone functions.*

The lemma above provides us with the desired approximation guarantees for DCGreedy, and thus it remains to show the consistency property. Before doing so, we must address how the weights are computed on line 5 of the GreedyStep algorithm (see Algorithm 11). Computing the weights exactly requires exponential time, but they can be approximated in polynomial time using random samples. In order to illustrate the main ideas behind the proof of consistency, *we assume that the weights are computed exactly*, since this will keep the algorithm deterministic. In Sections 5.5 and 5.6, we remove this assumption and we analyze the resulting randomized algorithm using an extension of our framework.

Lemma 5.3.2. *Let A and B be two disjoint subsets of V. Suppose that, for each element $e \in B$, we have* DCGreedyRel($A \cup \{e\}$) = DCGreedyRel(A). *Then* DCGreedySol($A \cup B$) = DCGreedySol(A).

Proof. We will show that the GreedyStep algorithm picks the same set W on input $(A, \mathbf{x})$ and $(A \cup B, \mathbf{x})$, which implies the lemma. Suppose for contradiction that the algorithm makes different choices on input $(A, \mathbf{x})$ and $(A \cup B, \mathbf{x})$. Consider the first iteration where the two runs differ, and let e be the element added to W in that iteration on input $(A \cup B, \mathbf{x})$. Note that $e \notin A$ and thus we have $e \in B$. But then e will be added to W on input $(A \cup \{e\}, \mathbf{x})$. Thus $e \in \mathsf{DCGreedyRel}(A \cup \{e\})$, which contradicts the fact that $e \in B$. □

Thus we can apply the result from Section 5.1 and obtain the following result.

Theorem 5.3.3. *Let $f : 2^V \to \mathbb{R}_+$ be a submodular function, and $\mathcal{I} \subseteq 2^V$ be a matroid. For any $\epsilon > 0$ there is a randomized distributed $O(1/\epsilon)$-round algorithm that can be implemented in the MapReduce model (described in Section 2.4). The algorithm is an $(\alpha - O(\epsilon))$-approximation with constant probability for the problem $\max_{S \in \mathcal{I}} f(S)$, where α is $(1 - 1/e)$ for monotone f and $1/e$ for general f.*

5.4 Analysis of Theorem 5.1.3

We devote this section to the proof of Theorem 5.1.3.

Theorem 5.1.3. ParallelAlg *achieves an $(1 - \epsilon)^3\alpha$ approximation with constant probability.*

Proof. Let $R = 7/\epsilon$ be the total number of runs, and $\mathcal{C} = (C_0, C_1, \ldots, C_R)$. Let $I_r(C_{r-1}) \in \{0, 1\}$ be an indicator variable equal to 1 if and only if

$$\mathbf{E}_{X_{1,r}}[f(\mathsf{AlgSol}(C_{r-1} \cup X_{1,r}))] \geq (1 - \epsilon)^2\alpha \cdot f(\mathrm{OPT}).$$

In other words, $I_r(C_{r-1})$ is 1 if and only if the expected value of the solution obtained on a machine is a $(\alpha - O(\epsilon))$-approximation.

We now define variables tracking the progress of the algorithm. Let

$$\Phi_r(\mathcal{C}) = I_r(C_{r-1}) + \frac{2(f(C_r \cap \mathrm{OPT}) - f(C_{r-1} \cap \mathrm{OPT}))}{\epsilon f(\mathrm{OPT})},$$
$$\Phi(\mathcal{C}) = \sum_{r=1}^{R} \Phi_r(\mathcal{C}) \leq \sum_{r=1}^{R} I_r(C_{r-1}) + \frac{2f(C_R \cap \mathrm{OPT})}{\epsilon f(\mathrm{OPT})} \leq \sum_{r=1}^{R} I_r(C_{r-1}) + \frac{2}{\epsilon}.$$

Taking expectation over the random choices of $\mathcal{C}$, we have

$$\mathbf{E}_{\mathcal{C}}[\Phi(\mathcal{C})] \leq \sum_{r=1}^{R} \mathbf{E}[I_r(C_{r-1})] + \frac{2}{\epsilon}$$

On the other hand, by Theorem 5.1.2, $\mathbf{E}[\Phi_r(\mathcal{C})] \geq 1$ and therefore $\mathbf{E}[\Phi(\mathcal{C})] \geq R$. Thus

$$\frac{2}{\epsilon} + \sum_{r=1}^{R} \mathbf{E}[I_r(C_{r-1})] \geq \Phi(\mathcal{C}) \geq R.$$

Since $R > 6/\epsilon$, we have

$$\sum_{r=1}^{R} \mathbf{E}[I_r(C_{r-1})] \geq \frac{2R}{3}.$$

Therefore, with probability at least $2/3$, there exists a run r such that $I_r(C_{r-1}) = 1$. Fix the randomness up to the first such run, i.e., condition on a fixed $C_{r-1} = \widehat{C}_{r-1}$ such that $I_r(\widehat{C}_{r-1}) = 1$ and $C_r, \ldots, C_R$ remain random. Assume for contradiction that with probability at least $1 - \epsilon\alpha(1-\epsilon)^2$ over the choices of $X_{1,r}$,

$$f(\mathsf{AlgSol}(C_{r-1} \cup X_{1,r})) < (1-\epsilon)^3\alpha \cdot f(\mathrm{OPT}).$$

Then we have

$$\begin{aligned}
\mathbf{E}[f(\mathsf{AlgSol}(C_{r-1} \cup X_{1,r}))] &< (\epsilon\alpha(1-\epsilon)^2 + (1-\epsilon\alpha(1-\epsilon)^2)(1-\epsilon)^3\alpha) f(\mathrm{OPT}) \\
&= \left(\epsilon + (1-\epsilon\alpha(1-\epsilon)^2)(1-\epsilon)\right)(1-\epsilon)^2\alpha f(\mathrm{OPT}) \\
&< (1-\epsilon)^2\alpha f(\mathrm{OPT}),
\end{aligned}$$

contradicting our assumption on C_{r-1}. Thus, with probability at least $\epsilon\alpha(1-\epsilon)^2$, we have

$$f(\mathsf{AlgSol}(C_{r-1} \cup X_{1,r})) \geq (1-\epsilon)^3\alpha \cdot f(\mathrm{OPT}).$$

Notice that the above argument applies not only to machine 1 in run r but also the first machine in each of the g groups in the same run r and their random samples $X_{i,r}$ are independent. Thus, since $g \geq c/(\epsilon\alpha)$ for a sufficiently large constant c, with probability at least $5/6$, we have $\max_i f(\mathsf{AlgSol}(C_{r-1} \cup X_{i,r})) \geq (1-\epsilon)^3\alpha \cdot f(\mathrm{OPT})$. Overall, the algorithm succeeds with probability at least $2/3 \cdot 5/6 = 5/9$. □

5.5 A framework for parallelizing randomized algorithms

In this section, we extend the framework from Section 5.1 to the setting in which the sequential algorithm Alg is randomized.

We represent the randomness of Alg as a vector $\mathbf{b} \sim \mathcal{D}$ drawn from some distribution $\mathcal{D}$. It is convenient to have the randomness $\mathbf{b}$ given as input to the algorithm. More

precisely, we assume that Alg takes as input a subset $N \subseteq V$ and a random vector $\mathbf{b} \sim \mathcal{D}$ and returns a pair of sets, $\mathsf{AlgSol}(N, \mathbf{b})$ and $\mathsf{AlgRel}(N, \mathbf{b})$. We assume that the size of $\mathbf{b}$ depends only on the size of V, and hence is independent of the size of N. Finally, we assume that Alg has the following properties.

1. ($(\alpha, \epsilon, \delta)$-Approximation) Let $\mathrm{OPT} = \mathrm{argmax}_{S \in \mathcal{I}, S \subseteq V} f(S)$ be an optimal solution over the entire ground set V. Let $A \subseteq V$ and $\mathbf{b} \sim \mathcal{D}$. Let $B \subseteq \mathrm{OPT}$ be a subset such that, for each $e \in B$, $e \notin \mathsf{AlgRel}(A \cup \{e\}, \mathbf{b})$. We have

$$\Pr_{\mathbf{b} \sim \mathcal{D}} \left[f(\mathsf{AlgSol}(A \cup B, \mathbf{b})) \geq \alpha \cdot f((A \cup B) \cap \mathrm{OPT}) - \epsilon f(\mathrm{OPT}) \right] \geq 1 - \delta,$$

   where $\epsilon = \Omega(\alpha\delta)$.

2. (Consistency) Let $\mathbf{b}$ be any fixed vector. Let A and B be two disjoint subsets of V. Suppose that, for each element $e \in B$, we have $\mathsf{AlgRel}(A \cup \{e\}, \mathbf{b}) = \mathsf{AlgRel}(A, \mathbf{b})$. Then $\mathsf{AlgSol}(A \cup B, \mathbf{b}) = \mathsf{AlgSol}(A, \mathbf{b})$.

Note that our assumption that the length of $\mathbf{b}$ is independent of the size of the input subset allows expressions such as $\mathsf{Alg}(A \cup \{e\}, \mathbf{b})$ and $\mathsf{Alg}(A, \mathbf{b})$ to both make sense despite the fact that $|A \cup \{e\}| \neq |A|$. For simplicity, in the following we assume $\epsilon \geq \alpha\delta$.

Our algorithm works exactly as that described in Section 5.1, with the exception that each machine i in round r now additionally samples a random vector $\mathbf{b}_{i,r} \sim \mathcal{D}$. Then, on each machine, we run Alg on the set $V_{i,r} \coloneqq X_{i,r} \cup C$ of elements on the machine and obtain $\mathsf{AlgSol}(V_{i,r}, \mathbf{b}_{i,r})$ and $\mathsf{AlgRel}(V_{i,r}, \mathbf{b}_{i,r})$. As in Section 5.1, the union $\bigcup_i \mathsf{AlgRel}(V_{i,r}, \mathbf{b}_{i,r})$ of relevant elements is added to C, and the solution S_{best} is replaced by the best solution among $\{\mathsf{AlgSol}(V_{i,r}, \mathbf{b}_{i,r}) \colon 1 \leq i \leq M\}$ and S_{best}.

In the final round we place C on a single machine, sample a random vector $\mathbf{b} \sim \mathcal{D}$, and run Alg on C, and $\mathbf{b}$ to obtain the solution $\mathsf{AlgSol}(C, \mathbf{b})$. The final solution is the best among $\mathsf{AlgSol}(C, \mathbf{b})$ and S_{best}.

Analysis. The number of rounds, number of machines, space per machine, and amount of communication are the same as in Section 5.1. Thus, we focus on the approximation guarantee of the parallel algorithm. Using Theorem 5.5.1 instead of Theorem 5.1.2, we can then finish the analysis in almost the same way as the deterministic case. The only difference is that instead of arguing that the algorithm works well with most of the random choices for $X_{i,r}$ as before, the proof now argues that the algorithm works well

with most of the random choices for both $X_{i,r}$ and $\mathbf{b}_{i,r}$. Nonetheless, the same proof except for this substitution works.

Theorem 5.5.1. *Consider a run $r > 1$ of the algorithm. Let $\widehat{C}_{r-1} \subseteq V$. Then one of the following must hold:*

(1) $\mathbf{E}_{X_{1,r},\mathbf{b}_{1,r}}[f(\mathsf{AlgSol}(C_{r-1} \cup X_{1,r}, \mathbf{b}_{1,r})) \mid C_{r-1} = \widehat{C}_{r-1}] \geq (\alpha - O(\epsilon)) \cdot f(\mathrm{OPT})$, *or*

(2) $\mathbf{E}[f(C_r \cap \mathrm{OPT}) \mid C_{r-1} = \widehat{C}_{r-1}] - f(\widehat{C}_{r-1} \cap \mathrm{OPT}) \geq \frac{\epsilon}{2} \cdot f(\mathrm{OPT})$.

Proof. Consider a run r of the algorithm. Let $\widehat{C}_{r-1} \subseteq V$. In the following, we condition on the event that $C_{r-1} = \widehat{C}_{r-1}$.

For each element $e \in V$, let $p_r(e) = \Pr_{X \sim \mathcal{V}(1/m), \mathbf{b} \sim \mathcal{D}}[e \in \mathsf{AlgRel}(\widehat{C}_{r-1} \cup X \cup \{e\}, \mathbf{b})]$, if $e \in \mathrm{OPT} \setminus \widehat{C}_{r-1}$, and 0 otherwise. The proof of the following lemma is exactly the same as Lemma 5.1.4 and thus is omitted.

Lemma 5.5.2. *For each element $e \in \mathrm{OPT} \setminus \widehat{C}_{r-1}$,*

$$\Pr[e \in \cup_{1 \leq i \leq gm} \mathsf{AlgRel}(\widehat{C}_{r-1} \cup X_{i,r}, \mathbf{b}_{i,r})] = 1 - (1 - p_r(e))^g,$$

where g is the number of groups into which the machines are partitioned.

We define a partition (P_r, Q_r) of $\mathrm{OPT} \setminus \widehat{C}_{r-1}$ as follows:

$$P_r = \{e \in \mathrm{OPT} \setminus \widehat{C}_{r-1} \colon p_r(e) < \epsilon\}, \qquad Q_r = \{e \in \mathrm{OPT} \setminus \widehat{C}_{r-1} \colon p_r(e) \geq \epsilon\}.$$

The following subsets of P_r and Q_r are key to our analysis (recall that $X_{i,r}$ is the random sample placed on machine i at the beginning of the run and $\mathbf{b}_{i,r}$ is the random vector sampled by machine i in round r):

$$P'_r = \{e \in P_r \colon e \notin \mathsf{AlgRel}(\widehat{C}_{r-1} \cup X_{1,r} \cup \{e\}, \mathbf{b}_{1,r})\},$$
$$Q'_r = Q_r \cap \left(\cup_{i=1}^{gm} \mathsf{AlgRel}(\widehat{C}_{r-1} \cup X_{i,r}, \mathbf{b}_{i,r}) \right).$$

Note that each element $e \in P_r$ is in P'_r with probability $1 - p_r(e) \geq 1 - \epsilon$. Further, by Lemma 5.5.2, each element $e \in Q_r$ is in Q'_r with probability $1 - (1 - p_r(e))^g \geq 1 - \frac{1}{e} \geq \frac{1}{2}$.

It follows from the definition of P'_r and the consistency property of Alg that

$$\mathsf{AlgSol}(\widehat{C}_{r-1} \cup X_{1,r}, \mathbf{b}_{1,r}) = \mathsf{AlgSol}(\widehat{C}_{r-1} \cup X_{1,r} \cup P'_r, \mathbf{b}_{1,r}). \tag{5.3}$$

Let $\mathrm{OPT}_{r-1} = \widehat{C}_{r-1} \cap \mathrm{OPT}$ be the part of OPT in this iteration's pool. We apply the $(\alpha, \epsilon, \delta)$-approximation property with $A = \widehat{C}_{r-1} \cup X_{1,r}$, $\mathbf{b} = \mathbf{b}_{1,r}$, and $B = P'_r$ to obtain

$$\Pr_{\mathbf{b}_{1,r}}\left[\mathsf{AlgSol}(\widehat{C}_{r-1} \cup X_{1,r} \cup P'_r, \mathbf{b}_{1,r}) \geq \alpha \cdot f((\widehat{C}_{r-1} \cup X_{1,r} \cup P'_r) \cap \mathrm{OPT}) - \epsilon f(\mathrm{OPT})\right] \geq 1 - \delta.$$

Since f is monotone when restricted to OPT (Lemma 2.2.1), and $P'_r \cup \mathrm{OPT}_{r-1} \subseteq (\widehat{C}_{r-1} \cup X_{1,r} \cup P'_r) \cap \mathrm{OPT}$, this inequality implies that

$$\Pr_{\mathbf{b}_{1,r}}\left[\mathsf{AlgSol}(\widehat{C}_{r-1} \cup X_{1,r} \cup P'_r, \mathbf{b}_{1,r}) \geq \alpha \cdot f(P'_r \cup \mathrm{OPT}_{r-1}) - \epsilon f(\mathrm{OPT})\right] \geq 1 - \delta.$$

Therefore, equation (5.3) gives

$$\Pr_{\mathbf{b}_{1,r}}\left[\mathsf{AlgSol}(\widehat{C}_{r-1} \cup X_{1,r}, \mathbf{b}_{1,r}) \geq \alpha \cdot f(P'_r \cup \mathrm{OPT}_{r-1}) - \epsilon f(\mathrm{OPT})\right] \geq 1 - \delta.$$

Taking expectation on both sides gives

$$\begin{aligned}
\mathbf{E}_{X_{1,r}, b_{1,r}}[f(\mathsf{AlgSol}(\widehat{C}_{r-1} \cup X_{1,r}, \mathbf{b}_{1,r}))] &\geq (1-\delta)\alpha \cdot \mathbf{E}_{X_{1,r}}[f(P'_r \cup \mathrm{OPT}_{r-1})] - \epsilon f(\mathrm{OPT}) \\
&\geq \alpha \cdot \mathbf{E}_{X_{1,r}}[f(P'_r \cup \mathrm{OPT}_{r-1})] - (\epsilon + \alpha\delta) f(\mathrm{OPT}) \\
&\geq (1-\epsilon)\alpha \cdot f(P_r \cup \mathrm{OPT}_{r-1}) - (\epsilon + \alpha\delta) f(\mathrm{OPT}) \\
&\geq (1-\epsilon)\alpha \cdot f(P_r \cup \mathrm{OPT}_{r-1}) - 2\epsilon f(\mathrm{OPT}).
\end{aligned} \tag{5.4}$$

Here, the second inequality follows from the fact that f is monotone restricted to $\mathrm{OPT} \supseteq (P'_r \cup \mathrm{OPT}_{r-1})$, the third from Lemma 2.3.2 and the fact that every element of P_r appears in P'_r with probability at least $(1-\epsilon)$, and the last from our assumption that $\alpha\delta \leq \epsilon$.

Next, note that $Q'_r \subseteq (C_r \cap \mathrm{OPT}) \setminus \mathrm{OPT}_{r-1}$. This together with monotonicity of f restricted to r imply:

$$\begin{aligned}
\mathbf{E}[f(C_r \cap \mathrm{OPT}) \mid C_{r-1} = \widehat{C}_{r-1}] &\geq \mathbf{E}[f(Q'_r \cup \mathrm{OPT}_{r-1})] \\
&\geq \frac{1}{2} \cdot f(Q_r \cup \mathrm{OPT}_{r-1}) + \frac{1}{2} \cdot f(\mathrm{OPT}_{r-1}),
\end{aligned}$$

where the last inequality follows from Lemma 2.3.2 and Lemma 5.5.2. Subtracting $f(\mathrm{OPT}_{r-1})$ from both sides, we get

$$\mathbf{E}[f(C_r \cap \mathrm{OPT}) \mid C_{r-1} = \widehat{C}_{r-1}] - f(\mathrm{OPT}_{r-1}) \geq \frac{1}{2}\left(f(Q_r \cup \mathrm{OPT}_{r-1}) - f(\mathrm{OPT}_{r-1})\right).$$

Using the condition $C_{r-1} = \widehat{C}_{r-1}$ and the definition $\mathrm{OPT}_{r-1} = \widehat{C}_{r-1} \cap \mathrm{OPT}$ we obtain:

$$\begin{aligned}
&\mathbf{E}[f(C_r \cap \mathrm{OPT}) - f(C_{r-1} \cap \mathrm{OPT}) \mid C_{r-1} = \widehat{C}_{r-1}] \\
&\qquad \geq \frac{1}{2}\left(f(Q_r \cup \mathrm{OPT}_{r-1}) - f(\mathrm{OPT}_{r-1})\right) \\
&\qquad \geq \frac{1}{2}\left(f(P_r \cup Q_r \cup \mathrm{OPT}_{r-1}) - f(P_r \cup \mathrm{OPT}_{r-1})\right) \\
&\qquad = \frac{1}{2}\left(f(\mathrm{OPT}) - f(P_r \cup (\widehat{C}_{r-1} \cap \mathrm{OPT}))\right), \qquad (5.5)
\end{aligned}$$

where the second inequality follows from submodularity.

Now, if $f(P_r \cup (\widehat{C}_{r-1} \cap \mathrm{OPT})) \geq (1-\epsilon) \cdot f(\mathrm{OPT})$ then, by (5.4), we have

$$\mathbf{E}[f(\mathsf{AlgSol}(\widehat{C}_{r-1} \cup X_{1,r}))] \geq (1-\epsilon)\alpha \cdot f(P_r \cup (\widehat{C}_{r-1} \cap \mathrm{OPT})) \geq (1-\epsilon)^2\alpha \cdot f(\mathrm{OPT}),$$

and the first property in the statement of Theorem 5.1.2 holds. Otherwise, $f(\mathrm{OPT}) - f(P_r \cup (\widehat{C}_{r-1} \cap \mathrm{OPT})) \geq \epsilon \cdot f(\mathrm{OPT})$, and we have from (5.5)

$$\begin{aligned}
&\mathbf{E}[f(C_r \cap \mathrm{OPT}) - f(C_{r-1} \cap \mathrm{OPT}) \mid C_{r-1} = \widehat{C}_{r-1}] \\
&\qquad \geq \frac{1}{2}\left(f(\mathrm{OPT}) - f(P_r \cup (\widehat{C}_{r-1} \cap \mathrm{OPT}))\right) \\
&\qquad \geq \frac{\epsilon}{2} \cdot f(\mathrm{OPT}),
\end{aligned}$$

which implies that the second property must hold.

□

5.6 Analysis of DCGreedy for the application of the randomized framework

In this section, we show that we can instantiate the randomized framework from Section 5.5 with a modified DCGreedy algorithm, and obtain the results stated in Section 5.3. Specifically, we extend the DCGreedy algorithm to the setting in which the weights w_e on line 5 of GreedyStep are evaluated approximately via samples.

The resulting GreedyStep is shown in Algorithm 12. Notice that it is essentially the same as Algorithm 11, with different computation of the weights on line 7. The weights w_e are estimated as follows. Given ℓ independent random sets $R_1, \ldots, R_\ell$ (the samples for $R(\mathbf{x}+\mathbf{y})$), w_e is set to $\frac{1}{\ell}\sum_{i=1}^{\ell}(f(R_i \cup \{e\}) - f(R_i))$.

We devote the rest of this section to proving the following result.

Algorithm 12: Greedy Update Step (GreedyStep) with randomized approximation of w_e's.

Input: $N \subseteq V$, $\mathbf{x} \in [0,1]^N$

1 $W \leftarrow \emptyset$
2 $\mathbf{y} \leftarrow 0$
3 **repeat**
4 Let $D \leftarrow \{e \in N \setminus W : W \cup \{e\} \in \mathcal{I}\}$
5 Pick $\ell = \Theta\left(\frac{s \log n}{\epsilon^2}\right)$ independent random samples for $R(\mathbf{x}+\mathbf{y})$
6 **foreach** $e \in D$ **do**
7 $w_e \leftarrow$ approximation of $\mathbf{E}[f(R(\mathbf{x}+\mathbf{y}) \cup \{e\}) - f(R(\mathbf{x}+\mathbf{y}))]$ via above samples
8 Let $e^* = \arg\max_{e \in D}\{w_e\}$
9 **if** $D = \emptyset$ *or* $w_{e^*} < 0$ **then**
10 **return y**
11 **else** $y_{e^*} \leftarrow y_{e^*} + \epsilon(1 - x_{e^*})$
12 $W \leftarrow W \cup \{e\}$

Figure 5.2: Discretized continuous greedy (DCGreedy) with approximate evaluation of the w_e's on line 7.

Theorem 5.6.1. *The modified* DCGreedy *algorithm with approximate evaluation of w_e's satisfies the consistency property and the $(\alpha, \epsilon, \delta)$-approximation property with $\delta = 1/n$ and $\alpha = 1/e - O(\epsilon)$ for non-monotone functions and $\alpha = 1 - 1/e - O(\epsilon)$ for monotone functions.*

We begin by verifying that the consistency property holds. Consider a vector $\mathbf{b}$ and two subsets $A, B \subseteq V$ such that, for each element $e \in B$, we have $\mathsf{DCGreedyRel}(A \cup \{e\}, \mathbf{b}) = \mathsf{DCGreedyRel}(A, \mathbf{b})$. We shall show that the approximate GreedyStep algorithm always picks the same set W on input $(A, \mathbf{x}, \mathbf{b})$ and $(A \cup B, \mathbf{x}, \mathbf{b})$. (Note that, since the two runs have the same randomness $\mathbf{b}$, they will use the same approximate weights.) Suppose for contradiction that the algorithm makes different choices on input $(A, \mathbf{x}, \mathbf{b})$ and $(A \cup B, \mathbf{x}, \mathbf{b})$. Consider the first iteration where the two runs differ, and let e be the element added to W in that iteration on input $(A \cup B, \mathbf{x}, \mathbf{b})$. Note that $e \notin A$ and thus we have $e \in B$. But then e would be added to W on input $(A \cup \{e\}, \mathbf{x}, \mathbf{b})$, as well. Thus $e \in \mathsf{DCGreedyRel}(A \cup \{e\}, \mathbf{b})$, which contradicts the fact that $e \in B$. Thus the consistency property holds.

Now we verify that the $(\alpha, \epsilon, \delta)$-approximation property holds. The analysis of the modified DCGreedy algorithm is similar to the analyses in [Feldman et al., 2011;

Badanidiyuru and Vondrák, 2014]. In the following, F is the multilinear extension of f, defined in Section 2.3.2.

Lemma 5.6.2. *Let $\mathcal{I}$ be matroid on V and $\mathrm{OPT} = \mathrm{argmax}_{S\in\mathcal{I},S\subseteq V} f(S)$. Let $A \subseteq V$ and $\mathbf{b} \sim \mathcal{D}$. Let $B \subseteq \mathrm{OPT}$ be a subset such that, for each $e \in B$, $e \notin \mathsf{DCGreedy}(A \cup \{e\}, \mathbf{b})$. Then, we have*

$$\Pr_{\mathbf{b}\sim\mathcal{D}}[F(\mathsf{DCGreedy}(A \cup B, \mathbf{b})) \geq \alpha \cdot f((A \cup B) \cap \mathrm{OPT}) - \epsilon \cdot f(\mathrm{OPT})] \geq 1 - 1/n,$$

where $\alpha = (1 - 1/e - O(\epsilon))$ for monotone f and $(1/e - O(\epsilon))$ for general f.

In the remainder of this section, we prove Lemma 5.6.2. Let $\mathrm{OPT} = \mathrm{argmax}_{S\in I} f(S)$ be an optimal solution over the entire ground set V, and consider the execution of $\mathsf{DCGreedy}(A \cup \mathrm{OPT}, \mathbf{b})$. Let Z be the set of vectors that $\mathsf{DCGreedy}(A \cup \mathrm{OPT}, \mathbf{b})$ considers when computing the weights of elements, i.e., the set of all vectors $\mathbf{z} \coloneqq \mathbf{x} + \mathbf{y}$, where $\mathbf{x} = \mathbf{x}(t)$ for some iteration t of $\mathsf{DCGreedy}(A \cup \mathrm{OPT}, \mathbf{b})$ and $\mathbf{y}$ is the vector constructed by previous iterations of $\mathsf{GreedyStep}(A \cup \mathrm{OPT}, \mathbf{x}, \mathbf{b})$. Formally, we associate the vector $\mathbf{z}_j \in Z$ with the jth execution of $\mathsf{GreedyStep}$'s main loop (counted across all the iterations of $\mathsf{DCGreedy}$). Note that $|Z| \leq s/\epsilon$, since $\mathsf{GreedyStep}$'s loop is executed at most s times for each of the $1/\epsilon$ iterations of $\mathsf{DCGreedy}$. For each sample, the random string $\mathbf{b}$ can simply store $|V|$ random thresholds in $[0, 1]$. For a given vector $\mathbf{z}$, these thresholds can be used to round $\mathbf{z}$ to an integral indicator vector (a sample of $R(\mathbf{z})$) in order to estimate $\mathbf{E}[f(R(\mathbf{z}) \cup \{e\}) - f(R(\mathbf{z})]$.

Consider the jth time $\mathsf{GreedyStep}$ executes line 7, and suppose that for each element $e \in A \cup \mathrm{OPT}$ we compute a weight $w_e(\mathbf{z}_j, \mathbf{b})$, by using ℓ random samples encoded by $\mathbf{b}$ to estimate $R(\mathbf{z}_j)$, as in $\mathsf{GreedyStep}$. We say that $w_e(\mathbf{z}_j, \mathbf{b})$ is a good estimate if

$$|w_e(\mathbf{z}_j, \mathbf{b}) - \mathbf{E}[f(R(\mathbf{z}_j) \cup \{e\}) - f(R(\mathbf{z}_j))]| \leq \frac{\epsilon}{2s} f(\mathrm{OPT}) + \frac{\epsilon}{2}\,\mathbf{E}[f(R(\mathbf{z}_j) \cup \{e\}) - f(R(\mathbf{z}_j))].$$

We say that $\mathbf{b}$ is good if all of the weights $\{w_e(\mathbf{z}_j, \mathbf{b}) \colon \mathbf{z}_j \in Z, e \in A \cup \mathrm{OPT}\}$ are good estimates.

Lemma 5.6.3. *The randomness $\mathbf{b}$ is good with probability at least $1 - 1/n$.*

Proof. Let $d = \max_{e\in V} f(e) \leq f(\mathrm{OPT})$. Consider a weight w_e and let $R_1, \ldots, R_\ell$ denote the independent random sets used to compute w_e in line 7 of $\mathsf{GreedyStep}$. For each $i \in [\ell]$, let $w_{e,i} = f(R_i \cup \{e\}) - f(R_i)$. Note that, by submodularity, $w_{e,i} \leq d \leq f(\mathrm{OPT})$. We use the following version of the Chernoff bound.

Lemma 5.6.4 (Badanidiyuru and Vondrák [2014], Lemma 2.3)**.** *Let $X_1, \ldots, X_m$ be independent random variables such that for each i, $X_i \in [0,1]$. Let $X = \frac{1}{m}\sum_{i=1}^{m} X_i$ and $\mu = \mathbf{E}[X]$. Then*

$$\Pr[X > (1+\alpha)\mu + \beta] \leq \exp\left(-\frac{m\alpha\beta}{3}\right),$$
$$\Pr[X < (1-\alpha)\mu - \beta] \leq \exp\left(-\frac{m\alpha\beta}{2}\right).$$

If we choose an appropriately large value in the definition of ℓ (recall we set $\ell = \Theta\left(\frac{s \log n}{\epsilon^2}\right)$ in Algorithm 12), then setting $m = \ell$, $X_i = w_{e,i}/d$, $\alpha = \epsilon/2$, and $\beta = \epsilon/2s$ in Lemma 5.6.4, we obtain that w_e is a good estimate with probability at least $1 - 1/n^4 \geq 1 - \epsilon/(sn^2)$. The size of Z is at most s/ϵ and for each element of Z, there are at most n weights to be estimated, so the lemma follows by the union bound. □

Now, note that if some random string $\mathbf{b}$ is good, then all weights calculated by $\mathsf{DCGreedy}(A \cup B, \mathbf{b})$ are good also, since $A \cup B \subseteq A \cup \mathrm{OPT}$, and, as we have noted, the consistency property implies that $\mathsf{GreedyStep}$ picks the same set on inputs $(A \cup B, \mathbf{x}, \mathbf{b})$ and $(A, \mathbf{x}, \mathbf{b})$ in each iteration. We now fix some good $\mathbf{b}$, and show that for *any* $B \subseteq \mathrm{OPT} \setminus A$, we must have:

$$F(\mathsf{DCGreedy}(A \cup B, \mathbf{b})) \geq \alpha \cdot f((A \cup B) \cap \mathrm{OPT}). \tag{5.6}$$

Where $\alpha = 1/e - O(\epsilon)$ for non-monotone functions and $\alpha = 1 - 1/e - O(\epsilon)$ for monotone functions. When f is monotone, this follows from previous work [Badanidiyuru and Vondrák, 2014]. Thus we focus on the non-monotone case. This will finish the proof Lemma 5.6.2.

Let $N = A \cup B$ for some $B \subseteq \mathrm{OPT} \setminus A$ and consider the restricted maximization problem $\max_{S \subseteq N, S \in \mathcal{I}} f(S)$.

Now, we begin by showing that $\mathsf{DCGreedy}$ improves the current solution by a large amount in each step.

Lemma 5.6.5. *Suppose that the randomness $\mathbf{b}$ is good. In each iteration t of* $\mathsf{DCGreedy}$*, $F(\mathbf{x}(t)) - F(\mathbf{x}(t-1)) \geq \epsilon(1-\epsilon)((1-\epsilon)^t f(N \cap \mathrm{OPT}) - F(\mathbf{x}(t))) - \epsilon^2 f(\mathrm{OPT})$.*

Proof. Fix an iteration t, and for brevity denote $\mathbf{x} = \mathbf{x}(t-1)$, $\mathbf{x}' = \mathbf{x}(t)$. Let W be the set of elements selected by the $\mathsf{GreedyStep}$ for this update, and let $\mathbf{y}$ be the associated update vector. We suppose without loss of generality that $|W| = s$, where s is the rank

of the matroid $\mathcal{I}$, since if $|W| < s$ we can simply add $s - |W|$ dummy elements to W. Let e_i be the ith element added to W by GreedyStep and let $\mathbf{y}(i)$ be the value of $\mathbf{y}$ after i elements have been added to W.

By Lemma 2.1.7, there is a bijective mapping $\pi : N \cap \mathrm{OPT} \to W'$ between $N \cap \mathrm{OPT}$ and a subset $W' \subseteq W$ of size $|N \cap \mathrm{OPT}|$ such that, for each element $o \in N \cap \mathrm{OPT}$, $W \setminus \{\pi(o)\} \cup \{o\} \in \mathcal{I}$. For each $i \in [s]$, let $o_i \coloneqq \pi^{-1}(e_i)$ if $e_i \in W'$ and $o_i \coloneqq e_i$ otherwise.

For each i, we have $w_{e_i} \geq w_{o_i}$, since o_i is a candidate element during the iteration of GreedyStep that picked e_i. Thus, since all the weights are good estimates, we have

$$\begin{aligned}&\mathbf{E}[f(R(\mathbf{x}+\mathbf{y}(i-1)) \cup \{e_i\}) - f(R(\mathbf{x}+\mathbf{y}(i-1)))] \\ &\quad \geq (1-\epsilon)\,\mathbf{E}[f(R(\mathbf{x}+\mathbf{y}(i-1)) \cup \{o_i\}) - f(R(\mathbf{x}+\mathbf{y}(i-1)))] - \frac{\epsilon}{s} f(\mathrm{OPT}). \quad (5.7)\end{aligned}$$

for all $\mathbf{y}$ and i.

Then, we have:

$$\begin{aligned}&F(\mathbf{x}') - F(\mathbf{x}) = F(\mathbf{x}+\mathbf{y}) - F(\mathbf{x}) \\ &\quad = \sum_{i=1}^{s} (F(\mathbf{x}+\mathbf{y}(i)) - F(\mathbf{x}+\mathbf{y}(i-1))) \\ &\quad = \sum_{i=1}^{s} \epsilon(1-x_{e_i}) \frac{\partial F}{\partial x_{e_i}}\Big|_{\mathbf{x}+\mathbf{y}(i-1)} \\ &\quad = \sum_{i=1}^{s} \epsilon\,\mathbf{E}[f(R(\mathbf{x}+\mathbf{y}(i-1)) \cup \{e_i\}) - f(R(\mathbf{x}+\mathbf{y}(i-1)))] \\ &\quad \geq \sum_{i=1}^{s} \epsilon \left((1-\epsilon)\,\mathbf{E}[f(R(\mathbf{x}+\mathbf{y}(i-1)) \cup \{o_i\}) - f(R(\mathbf{x}+\mathbf{y}(i-1)))] - \frac{\epsilon}{s} f(\mathrm{OPT})\right) \\ &\quad \geq \sum_{i=1}^{s} \epsilon \left((1-\epsilon)\,\mathbf{E}[f(R(\mathbf{x}') \cup \{o_i\}) - f(R(\mathbf{x}'))] - \frac{\epsilon}{s} f(\mathrm{OPT})\right) \\ &\quad \geq \epsilon(1-\epsilon)(F(\mathbf{x}' \vee \mathbf{1}_{N \cap \mathrm{OPT}}) - F(\mathbf{x}')) - \epsilon^2 f(\mathrm{OPT}), \quad (5.8)\end{aligned}$$

where the first inequality follows from (5.7) and the last two from submodularity.

We relate the value $F(\mathbf{x}' \vee \mathbf{1}_{N \cap \mathrm{OPT}})$ to $f(\mathrm{OPT})$ using Lemma 2.3.4. At each step, we increase each coordinate e of $\mathbf{x}$ by at most $\epsilon(1 - x_e(t))$. Thus, for any step $0 \leq j \leq 1/\epsilon$, we have

$$x_e(j+1) - x_e(j) \leq (1 - x_e(j))\epsilon,$$

or, equivalently,

$$x_e(j+1) - (1-\epsilon)x_e(j) \leq \epsilon.$$

Thus, for each time step $t \leq 1/\epsilon$, we have

$$x_e(t) \leq \sum_{j=1}^{t} \epsilon(1-\epsilon)^{t-1-j} = 1 - (1-\epsilon)^t$$

By combining the inequality above with Lemma 2.3.4, we obtain

$$F(\mathbf{x}(t) \vee \mathbf{1}_{N \cap \mathrm{OPT}}) \geq (1-\epsilon)^t f(N \cap \mathrm{OPT}).$$

Plugging this bound into (5.8) then completes the proof. □

Lemma 5.6.6. *Suppose that the randomness* $\mathbf{b}$ *is good. The final solution* $\mathbf{x}(1/\epsilon)$ *constructed by* $\mathsf{DCGreedy}(N, \mathbf{b})$ *satisfies* $F(\mathbf{x}(1/\epsilon)) \geq (1/e - \epsilon)f(N \cap \mathrm{OPT}) - \epsilon f(\mathrm{OPT})$. *Therefore the integral solution* S *satisfies* $f(S) \geq (1/e - \epsilon)f(N \cap \mathrm{OPT}) - \epsilon f(\mathrm{OPT})$.

Proof. By rearranging the inequality from Lemma 5.6.5, we obtain

$$\begin{aligned} F(\mathbf{x}(t)) &\geq \frac{\epsilon(1-\epsilon)^{t+1} f(N \cap \mathrm{OPT}) + F(\mathbf{x}(t-1)) - \epsilon^2 f(\mathrm{OPT})}{1+\epsilon} \\ &\geq \epsilon(1-\epsilon)^{t+2} f(N \cap \mathrm{OPT}) + (1-\epsilon)F(\mathbf{x}(t-1)) - \epsilon^2 f(\mathrm{OPT}). \end{aligned}$$

It follows by induction that

$$F(\mathbf{x}(t)) \geq t\epsilon(1-\epsilon)^{t+2} f(N \cap \mathrm{OPT}) - t\epsilon^2 f(\mathrm{OPT}).$$

Thus

$$F(\mathbf{x}(1/\epsilon)) \geq (1-\epsilon)^{\frac{1}{\epsilon}+2} f(N \cap \mathrm{OPT}) - \epsilon f(\mathrm{OPT}) \geq \left(\frac{1}{e} - \epsilon\right) f(N \cap \mathrm{OPT}) - \epsilon f(\mathrm{OPT}).$$ □

Combining Lemmas 5.6.3 and 5.6.6 then complete the proof of Lemma 5.6.2.

5.7 Two-round algorithm for monotone submodular maximization with a cardinality constraint

We now show how the previous techniques give a simple two-round algorithm that achieves a $1/2 - \epsilon$ approximation for monotone maximization subject to a cardinality constraint.

Algorithm 13: Two-round algorithm for monotone submodular maximization subject to cardinality constraint

Input: instance $\langle V, \mathcal{I}, f \rangle$, number g of groups, number m of machines, desired accuracy ϵ

for *each group of machines* **do**
  for $e \in V$ **do**
    Assign e to a machine i chosen uniformly at random
Let V_i be the elements assigned to machine i
Run $\mathsf{Greedy}(V_i)$ on each machine i to obtain S_i
Place $S = (\bigcup_i S_i) \cup X$ on machine 1, where X is a random sample of V
for $a \in \{0, 1, \ldots, k\}$ **do**
  $T_a^1 \leftarrow \mathsf{Greedy}(f, X, a)$
  $T_a^2 \leftarrow \mathsf{Greedy}(g, S, k - a)$, where $g(A) = f(T_a^1 \cup A) - f(T_a^1)$
  $T_a \leftarrow T_a^1 \cup T_a^2$
return $\arg\max_{0 \le a \le k} \{f(T_a)\}$

Theorem 5.7.1. *There is a randomized, two-round, distributed algorithm achieving a* $\frac{1}{2} - \epsilon$ *approximation (in expectation) for* $\max_{S:|S| \le k} f(S)$*, where* f *is a monotone submodular function.*

Although it presents an approximation factor worse than the one obtained by Mirrokni and Zadimoghaddam [2015], our algorithm is simpler to implement, and uses smaller core-sets (size k, compared to $(2\sqrt(2) + 1)k$ in [Mirrokni and Zadimoghaddam, 2015]) and, thus, a smaller amount of communication overall.

5.7.1 Algorithm

Let $\epsilon > 0$ be a parameter. The algorithm uses $\Theta(\log(1/\epsilon)/\epsilon)$ groups of machines with m machines in each group (and thus the total number of machines is $O(m \log(1/\epsilon)/\epsilon)$).

We randomly distribute the ground set V to the machines as follows. Amongst each group of m machines, we partition V uniformly at random; each element e chooses an index $i \in [m]$ uniformly and independently at random and is assigned to the ith machine in the group. We do this separately for each group of machines, i.e., each element appears on exactly one machine in each group.

We run Greedy on each of the machines to select a set of k elements. Let S be the union of all of the Greedy solutions. We place S on a single machine together with a random sample $X \sim \mathcal{V}(1/m)$. On this machine, we pick the final solution as follows. For each value $a \in \{0, 1, \ldots, k\}$, we select a solution T_a as follows. Let $\mathsf{Greedy}(f, X, a)$ denote the first a elements chosen from the random sample X using the greedy algorithm on objective function f.

Then, let $T_a^1 = \mathsf{Greedy}(f, X, a)$ and define $g(A) = f(T_a^1 \cup A) - f(T_a^1)$ for each $A \subseteq V$. Note that g is a non-negative, monotone submodular function. Let $T_a^2 = \mathsf{Greedy}(g, S, k-a)$; that is, we pick $k-a$ elements from S using the Greedy algorithm with the function g as input. We set $T_a = T_a^1 \cup T_a^2$. The final solution T is the better of the $k+1$ solutions T_a, where $a \in \{0, 1, \ldots, k\}$.

5.7.2 Analysis

In the following, we show that the algorithm above is a $1/2 - \epsilon$ approximation. As in Section 4.2.2, for each element e, we define a probability $p_e = \Pr_{A \sim \mathcal{V}(1/m)}[e \in \mathsf{Greedy}(A \cup \{e\})]$, if $e \in \mathrm{OPT}$ and 0 otherwise. We define a partition (O_1, O_2) of OPT as follows:

$$O_1 = \{e \in \mathrm{OPT} \mid p_e < \epsilon\}, \qquad O_2 = \{e \in \mathrm{OPT} \mid p_e \geq \epsilon\}.$$

Let $a = |O_1|$ and let

$$O_1' = \{e \in O_1 \mid e \notin \mathsf{Greedy}(f, X \cup \{e\}, a)\}.$$

By the consistency property of the greedy algorithm (Lemma 4.1.1),

$$T_a^1 = \mathsf{Greedy}(f, X, a) = \mathsf{Greedy}(f, X \cup O_1', a).$$

Additionally, for a cardinality constraint, Greedy satisfies (GP) with $\gamma = 1/2$ (see Section 3.1.1). Therefore

$$f(T_a^1) \geq \frac{1}{2} f(T_a^1 \cup O_1'), \tag{5.9}$$

$$g(T_a^2) \geq \frac{1}{2} g(T_a^2 \cup (O_2 \cap S)). \tag{5.10}$$

The inequality (5.10) can be rewritten as

$$f(T_a^1 \cup T_a^2) - f(T_a^1) \geq \frac{1}{2}(f(T_a^1 \cup T_a^2 \cup (O_2 \cap S)) - f(T_a^1)). \tag{5.11}$$

Adding (5.9) and (5.11), we obtain

$$\begin{aligned} f(T_a) &\geq \frac{1}{2}(f(T_a^1 \cup O_1') + f(T_a^1 \cup T_a^2 \cup (O_2 \cap S)) - f(T_a^1)) \\ &\geq \frac{1}{2} f(T_a \cup O_1' \cup (O_2 \cap S)) \\ &\geq \frac{1}{2} f(O_1' \cup (O_2 \cap S)), \end{aligned}$$

where the last two inequalities follow from submodularity and monotonicity. Note that each element $e \in O_1$ is in O_1' with probability $1 - p_e \geq 1 - \epsilon$. Each element $e \in O_2$ is in the union of the Greedy solutions from a given group of machines with probability $p_e \geq \epsilon$; since there are $\Theta(\log(1/\epsilon)/\epsilon)$ groups of machines and the groups have independent partitions, e is in S with probability at least $1 - \epsilon$. Therefore

$$\mathbf{E}[\mathbf{1}_{O_1' \cup (O_2 \cap S)}] \geq (1 - \epsilon)\mathbf{1}_{\mathrm{OPT}}.$$

Thus

$$\mathbf{E}[f(T_a)] \geq \frac{1}{2} f^-(\mathbf{E}[\mathbf{1}_{O_1' \cup (O_2 \cap S)}]) \geq (1 - \epsilon)\frac{1}{2} f(\mathrm{OPT}).$$

In the last inequality, we have used that if $x \geq y$ component-wise and f is monotone, $f^-(x) \geq f^-(y)$.

Chapter 6

Parallel algorithms for unconstrained submodular maximization

In the present chapter we study strategies for designing parallel algorithms for submodular maximization in the unconstrained setting. To that end, we will use as ingredients the sequential algorithms presented in Chapter 3, namely, the Random Sample algorithm [Feige et al., 2011] (Section 3.3), and the Double Greedy algorithm [Buchbinder et al., 2012] (Section 3.4).

As demonstrated in Chapter 3, the Random Sample algorithm achieves an approximation factor of 1/4 by obliviously including each element in the solution with probability 1/2. On the other hand, the Double Greedy algorithm obtains a tight 1/2 approximation ratio by including elements in the solution with a probability that considers previous choices.

Despite the apparent differences both follow a similar structure: they consider each element i in the input set V exactly once, and, for each element, it is included in the solution with probability p_i, which may be conditioned on previous decisions, depending on the algorithm. These probabilities are given as follows.

- Random Sample algorithm: $p_i = 1/2$, for every i.
- Double Greedy algorithm: $p_i = [a_i]_+/([a_i]_+ + [b_i]_+)$, where $a_i = f(A_{i-1} + i) - f(A_{i-1})$, $b_i = f(B_{i-1} \setminus \{i\}) - f(B_{i-1})$.

One may then think of the two algorithms in a single "framework", shown in Algorithm 14.

As it achieves the best possible approximation guarantee for the problem, Double Greedy seemed a natural starting point for conceiving parallel algorithms for unconstrained submodular maximization problems. Pan et al. [2014] address the issue through the lens of parallel transaction processing systems (Section 2.4). In this case, the pro-

Algorithm 14: Generic algorithm.

Input: V, f

$A_0 \leftarrow \emptyset$
$B_0 \leftarrow V$
for $i \leftarrow 1$ *to* n **do**
    with probability p_i
    $A_i \leftarrow A_{i-1} \cup \{i\}$
    $B_i \leftarrow B_{i-1}$
    else $\langle\langle$*with probability* $1 - p_i\rangle\rangle$
    $A_i \leftarrow A_{i-1}$
    $B_i \leftarrow B_{i-1} \setminus \{i\}$
Return A_n

gram state kept at the server are the sets A and B, and the transactions are insertions or deletions to those. Constructing a transaction involves deciding which elements to include (evaluating marginal gains), while applying a transaction reduces to setting bits.

They propose two strategies, heavily inspired by the Double Greedy algorithm: *coordination-free*, which emphasizes speed at the cost of a weaker approximation ratio; and *concurrency control*, which guarantees the same 1/2 approximation of Double Greedy, at the cost of additional coordination and reduced parallelism. In the former, no transaction is failed at the client. Transactions are computed based on the information obtained when the element is pulled from the server, and committed to be processed. In the latter, guarantees are requested. If a transaction fails, the server must recompute the marginal gains involved in the Double Greedy algorithm before applying the changes serially to the program state.

We propose here a different approach for devising parallel algorithms for unconstrained submodular maximization problems in the same setting. We present two hybrid algorithms, combining Random Sample and Double Greedy steps, and analyze the trade-off between the approximation ratio attained and the degree of parallelism obtained.

We let t be a number of elements of the input set V ($t \in \{1, \ldots, n\}$), and let $p \coloneqq t/n$ be a fraction of the input set. Roughly speaking, both hybrid algorithms execute Double Greedy steps on t elements, and Random Sample steps on the remaining ones. The first, $\mathsf{Hybrid-I}$, runs Double Greedy on t elements, and then samples the remaining ones, achieving an approximation guarantee of $\max\left\{\frac{2+p}{8}, \frac{p}{2}\right\}$. In the second, $\mathsf{Hybrid-II}$, $n - t$ Random Sample steps are executed, and only then Double Greedy is performed on the last ones. It obtains an approximation ratio of $1/4 + p/8 + p^2/8$.

For simplicity, we present here the serial versions of the algorithms. In the context of parallel algorithms, we assume the Random Sample steps are performed in parallel without the need of guarantees, as in the coordination-free algorithm above, since the sampling is done independently. In fact, during the Random Sample stage, the elements may be split among the machines to be processed, and only afterwards the selected ones are communicated back to the server. The Double Greedy steps, on the other hand, require concurrency control, as in the second algorithm of Pan et al. [2014]. We can then see p as rough measure of parallelism – the larger p is, the less parallel the algorithm might be.

As in Section 3.3, for a set A, and $q \in [0, 1]$, we denote in the following by $A(q)$ the random set obtained by including each element in A independently at random with probability q.

6.1 Hybrid algorithm I

We consider the following hybrid algorithm ($\mathsf{Hybrid-I}$, Algorithm 15). We order the elements of V randomly and then run the Double Greedy algorithm for t steps to obtain sets A_t and B_t. Let $C_t = B_t \setminus A_t$ be the set of elements that were not considered by the Double Greedy algorithm. We execute Random Sample on C_t, and return the set $S \coloneqq A_t \cup C_t(1/2)$.

Note that our analysis works also for the following setting[1]: for each element, either: with probability p, it is processed serially by the server using Double Greedy; or, with probability $1 - p$, it is sent to one of the client machines, which will conduct a Random Sample step. At the end of processing, we return the solution obtained at the server together with all elements randomly selected by each machine.

We now turn to our analysis. We shall use the following lemma:

Lemma 6.1.1 (Filmus and Ward [2012], Lemma V.1)**.** *Let $f : 2^V \to \mathbb{R}_+$ be a submodular function, and let X be a set of t elements from V chosen uniformly at random. Then:*

$$\mathbf{E}[f(X)] \geq \frac{t}{|V|} \cdot f(V).$$

Our first technical lemma is the following:

Lemma 6.1.2. *Let $A_t \subseteq B_t \subseteq V$ be the sets obtained in the execution of Double Greedy steps in the $\mathsf{Hybrid-I}$ algorithm. Then, $\mathbf{E}[f(A_t) + f(B_t)] \geq t/n \cdot f(\mathrm{OPT})$.*

[1] In this setting, we must replace Lemma 6.1.1 with a similar lemma from Feige et al. [2011]. Also, the algorithm will then take pn oracle calls/communication rounds *in expectation.*

Algorithm 15: Hybrid algorithm I (Hybrid − I).

Input: V, f, t

Shuffle (V)
$A_0 \leftarrow \emptyset$
$B_0 \leftarrow V$
$C_0 \leftarrow V$
for $i \leftarrow 1$ *to* t **do**
  $\langle\langle$*Run* DoubleGreedy *for* i *from* 1 *to* $t\rangle\rangle$
  $C_i \leftarrow C_{i-1} \setminus \{i\}$
  $a_i \leftarrow f(A_{i-1} \cup \{i\}) - f(A_{i-1})$
  $b_i \leftarrow f(B_{i-1} \setminus \{i\}) - f(B_{i-1})$
  $p_i \leftarrow [a_i]_+ / ([a_i]_+ + [b_i]_+)$
  with probability p_i
  $A_i \leftarrow A_{i-1} \cup \{i\}$
  $B_i \leftarrow B_{i-1}$
  else $\langle\langle$*with probability* $1 - p_i\rangle\rangle$
  $A_i \leftarrow A_{i-1}$
  $B_i \leftarrow B_{i-1} \setminus \{i\}$
Return $A_t \cup C_t(1/2)$

Proof. Suppose that we run the Double Greedy algorithm to completion and let A be the (random) set that it produces. For any fixed set $A \subseteq V$ we can define the function:

$$\phi_A(S) = f(S \cap A) + f(V \setminus (S \setminus A)).$$

This function is submodular for any such A, and hence the function

$$\phi(S) = \mathbf{E}_A[\phi_A(S)],$$

is also submodular. Here, the expectation is over the random set A returned by the Double Greedy algorithm, or, equivalently, the random choices made by the Double Greedy algorithm. Suppose that we stop the Double Greedy algorithm after considering t elements $S_t \subseteq V$. Then, since the elements are ordered randomly, S_t is a t element

subset of V chosen uniformly at random. Let $p \coloneqq t/n$. Then,

$$\begin{aligned}
\mathbf{E}[f(A_t) + f(B_t)] &= \mathbf{E}[\phi(S_t)] \\
&\geq \mathbf{E}[p \cdot \phi(V)] && \text{(By Lemma 6.1.1)} \\
&= p \cdot \mathbf{E}[f(V \cap A) + f(V \setminus (V \setminus A))] \\
&= 2p \cdot \mathbf{E}[f(A)] \\
&\geq p \cdot f(OPT) && (\mathsf{DoubleGreedy} \text{ is a } \tfrac{1}{2}\text{-approximation})
\end{aligned}$$

□

Theorem 6.1.3. *Let $f : 2^V \to \mathbb{R}_+$ be a submodular function, let $t \in [n]$, and let parameter $p \coloneqq t/n$. The* $\mathsf{Hybrid-I}$ *algorithm is (in expectation) a $\left(\max\left\{\frac{2+p}{8}, \frac{p}{2}\right\}\right)$-approximation algorithm for $\max_{S \subseteq V} f(S)$.*

Proof. Let $S = A_t \cup C_t(1/2)$ be the output of the algorithm. Further, as in the Double Greedy analysis, let $\mathrm{OPT}_i \coloneqq (\mathrm{OPT} \cup A_i) \cap B_i$. Note that $C_t(1/2)$ is the union of a $1/2$ sample of $\mathrm{OPT}_t \setminus A_t$ and a $1/2$ sample of $(B_t \setminus A_t) \setminus \mathrm{OPT}_t$. Thus, it follows from Lemma 3.3.3 that

$$\begin{aligned}
\mathbf{E}[f(S)] &\geq \frac{1}{4}(\mathbf{E}[f(A_t)] + \mathbf{E}[\mathrm{OPT}_t] + \mathbf{E}[f(A_t \cup ((B_t \setminus A_t) \setminus \mathrm{OPT}_t))] + \mathbf{E}[f(B_t)]) \\
&\geq \frac{1}{4}(\mathbf{E}[f(A_t)] + \mathbf{E}[f(B_t)] + \mathbf{E}[f(\mathrm{OPT}_t)])
\end{aligned}$$

Hence,

$$\begin{aligned}
\mathbf{E}[f(S)] &\geq \frac{1}{4}\,\mathbf{E}[f(A_t) + f(B_t)] + \frac{1}{4}\,\mathbf{E}[f(OPT_t)] \\
&\geq \frac{1}{8}\,\mathbf{E}[f(A_t) + f(B_t)] + \frac{1}{4}\,\mathbf{E}[f(OPT_t)] + \frac{p}{8} f(OPT) && \text{(by Lemma 6.1.2)} \\
&\geq \frac{1}{4}\,(f(OPT) - \mathbf{E}[f(OPT_t)]) + \frac{1}{4}\,\mathbf{E}[f(OPT_t)] + \frac{p}{8} f(OPT) && \text{(by Lemma 3.4.2)} \\
&= \frac{1}{4} f(OPT) + \frac{p}{8} f(OPT).
\end{aligned}$$

On the other hand, we also have

$$\begin{aligned}
\mathbf{E}[f(S)] &= \mathbf{E}[f(A_t \cup C_t(1/2))] \\
&\geq \frac{1}{2}\,\mathbf{E}[f(A_t)] + \frac{1}{2}\,\mathbf{E}[f(B_t)] \\
&\geq \frac{p}{2} f(OPT) && \text{(by Lemma 6.1.2)}
\end{aligned}$$

Algorithm 16: Hybrid algorithm II ($\mathsf{Hybrid-II}$).

Input: V, f, p

$T \leftarrow V(1-p)$

$t' \leftarrow |T|$

$A_{t'} \leftarrow T(1/2)$

$B_{t'} \leftarrow V \setminus (T \setminus A_{t'})$

for $i \leftarrow t'+1$ *to* n **do**

  $\langle\langle$*Run* DoubleGreedy *for* i *from* $t'+1$ *to* $n\rangle\rangle$

  $a_i \leftarrow f(A_{i-1} \cup \{i\}) - f(A_{i-1})$

  $b_i \leftarrow f(B_{i-1} \setminus \{i\}) - f(B_{i-1})$

  $p_i \leftarrow [a_i]_+ / ([a_i]_+ + [b_i]_+)$

  with probability p_i

  $A_i \leftarrow A_{i-1} \cup \{i\}$

  $B_i \leftarrow B_{i-1}$

  else $\langle\langle$*with probability* $1-p_i\rangle\rangle$

  $A_i \leftarrow A_{i-1}$

  $B_i \leftarrow B_{i-1} \setminus \{i\}$

Return A_n

□

6.2 Hybrid algorithm II

Now, we consider a hybrid algorithm ($\mathsf{Hybrid-II}$, Algorithm 16) that runs the Random Sample algorithm followed by the Double Greedy. As before, fix a parameter $p \in [0,1]$. Let T be a subset of V in which each element appears independently with probability $1-p$. We run Random Sample on T and then Double Greedy on the remaining elements. Note that here, the number of elements t sent to Double Greedy is $n \cdot p$ *in expectation.*

Theorem 6.2.1. *Let $f : 2^V \to \mathbb{R}_+$ be a submodular function, and let parameter $p \in [0,1]$. The* $\mathsf{Hybrid-II}$ *algorithm proposed is (in expectation) a $\left(\frac{1}{4} + \frac{p}{8} + \frac{p^2}{8}\right)$-approximation algorithm for $\max_{S \subseteq V} f(S)$.*

Proof. For simplicity, we let $p' := 1-p$. Similarly to the analysis of Double Greedy, we consider the set:

$$\mathrm{OPT}_t = (\mathrm{OPT} \setminus T) \cup A_t,$$

which agrees with all A_t on all elements of T and with OPT on all elements of $V \setminus T$; that is, $A_t \subseteq \mathrm{OPT}_t \subseteq B_t$.

Consider an element $e \in \mathrm{OPT}$. We have:

$$\Pr[e \in \mathrm{OPT}_t] = 1 - \Pr[e \in T \setminus A_t] = 1 - \Pr[e \in T] \cdot \Pr[e \notin A_t \mid e \in T] = 1 - p'/2.$$

Next, suppose that $e \notin \mathrm{OPT}$. We have:

$$\Pr[e \in \mathrm{OPT}_t] = \Pr[e \in T \cap A_t] = \Pr[e \in T] \cdot \Pr[e \in A_t \mid e \in T] = p'/2.$$

Thus, we can view OPT_t as the union of the random sets $\mathrm{OPT}(1 - p'/2)$ and $(V \setminus \mathrm{OPT})(p'/2)$. Using Lemma 3.3.3, we get:

$$\begin{aligned}
\mathbf{E}[\mathrm{OPT}_t] &\geq \frac{p'}{2}\left(1 - \frac{p'}{2}\right) \cdot f(\emptyset) + \left(1 - \frac{p'}{2}\right)\frac{p'}{2} \cdot f(V \setminus \mathrm{OPT}) + \\
&\qquad \left(1 - \frac{p'}{2}\right)\left(1 - \frac{p'}{2}\right) \cdot f(\mathrm{OPT}) + \left(1 - \frac{p'}{2}\right)\frac{p'}{2} \cdot f(V) \\
&\geq \left(1 - \frac{p'}{2}\right)\left(1 - \frac{p'}{2}\right) \cdot f(\mathrm{OPT}) \\
&= \left(1 - p' + \frac{p'^2}{4}\right) \cdot f(\mathrm{OPT}). \qquad (6.1)
\end{aligned}$$

Now, we bound $\mathbf{E}[f(A_t) + f(B_t)]$. Conditioned on any choice of T, we have:

$$\begin{aligned}
\mathbf{E}[f(T(1/2))] + \mathbf{E}[f((V \setminus T) \cup T(1/2))] &\geq \frac{1}{4} \cdot f(\mathrm{OPT} \cap T) + \frac{1}{4} \cdot f((V \setminus T) \cup (\mathrm{OPT} \cap T)) \\
&= \frac{1}{4}\Big(f(\mathrm{OPT} \cap T) + f((V \setminus T) \cup \mathrm{OPT})\Big).
\end{aligned}$$

Taking the expectation over the choice of T, we then have:

$$\begin{aligned}
\mathbf{E}[f(A_t) + f(B_t)] &= \mathbf{E}[f(T(1/2))] + \mathbf{E}[f((V \setminus T) \cup T(1/2))] \\
&\geq \frac{1}{4}\,\mathbf{E}\left[f(\mathrm{OPT} \cap T) + f((V \setminus T) \cup \mathrm{OPT})\right] \\
&\geq \frac{1}{4}\Big(p' \cdot f(\mathrm{OPT}) + (1 - p')f(\emptyset) + p' \cdot f(\mathrm{OPT}) + (1 - p')f(V)\Big) \\
&\qquad\qquad \text{(by Lemma 3.3.2)} \\
&\geq \frac{p'}{2} \cdot f(\mathrm{OPT}). \qquad (6.2)
\end{aligned}$$

Now, suppose that we run the Double Greedy algorithm for $n - t$ steps, starting with sets A_t and B_t, to obtain a set A_n. We have $A_t \subseteq \mathrm{OPT}_t \subseteq B_t$, as in the Double Greedy analysis. It follows from Lemma 3.4.2 that, at each step i, we have:

$$\frac{1}{2}\,\mathbf{E}[f(A_i) + f(B_i) - f(A_{i-1}) - f(B_{i-1})] \geq \mathbf{E}[f(\mathrm{OPT}_{i-1}) - f(\mathrm{OPT}_i)].$$

Summing over steps $i = t + 1$ to $i = n$, we get:

$$\begin{aligned}\frac{1}{2}\,\mathbf{E}[f(A_n) + f(B_n) - (A_t) - f(B_t)] &\geq \mathbf{E}[f(\mathrm{OPT}_t) - f(\mathrm{OPT}_n)] \\ &= \mathbf{E}\left[f(\mathrm{OPT}_t)\right] - \frac{1}{2}\,\mathbf{E}\left[f(A_n) + f(B_n)\right],\end{aligned}$$

where the last equality follows since $\mathrm{OPT}_n = A_n = B_n$. That, together with (6.1) and (6.2) gives:

$$\begin{aligned}\mathbf{E}[f(A_n) + f(B_n)] &\geq \mathbf{E}\left[f(\mathrm{OPT}_t)\right] + \frac{1}{2}\,\mathbf{E}\left[f(A_t) + f(B_t)\right] \\ &\geq \left(1 - p' + \frac{p'^2}{4}\right) f(\mathrm{OPT}) + \frac{p'}{4} \cdot f(\mathrm{OPT}) \\ &= \left(1 + \frac{p'^2 - 3p'}{4}\right) f(\mathrm{OPT}).\end{aligned}$$

Thus, we have:

$$\begin{aligned}\mathbf{E}[f(A_n)] &\geq \left(\frac{1}{2} + \frac{p'^2 - 3p'}{8}\right) f(\mathrm{OPT}) \\ &\geq \left(\frac{1}{4} + \frac{p}{8} + \frac{p^2}{8}\right) f(\mathrm{OPT}).\end{aligned}$$

□

Chapter 7

Concluding remarks

This thesis presents new results in parallel algorithms for submodular maximization, both in the constrained and in the unconstrained settings. The contributions are summarized here and some possible future work directions are given in the following.

Building on [Mirzasoleiman et al., 2013], we devise in Chapter 4 distributed two-round algorithms based on Greedy. They achieve constant factor approximation guarantees under a variety of constraints, and work for both monotone and non-monotone functions. Moreover, we show *randomization* is a crucial part of the analysis.

In Section 4.2, we present the randomized counterpart of the GreeDI algorithm of Mirzasoleiman et al. [2013], RandGreeDi, and show that it achieves a $(\frac{\alpha}{2})$-approximation for monotone functions (Corollary 4.2.4), where α is the approximation guarantee of Greedy for a given constraint.

We then show that if the Greedy algorithm satisfies a stronger technical condition, the strong greedy property (Section 3.1.1), with a constant γ then our approach also lead to algorithms with constant approximation ratios for non-monotone submodular maximization. We propose in Section 4.3 a slightly modified version of RandGreeDi, NMRandGreeDi, and prove it obtains a $\frac{\gamma}{4+2\gamma}$ approximation (Corollary 4.3.4). Further, in Section 4.4, we show that the same RandGreeDi algorithm attains a $\left(\left(1-\frac{1}{m}\right)\frac{\beta\gamma}{\beta+\gamma}\right)$ approximation (Theorem 4.4.1) if Greedy is replaced for a β-approximation algorithm in the second round; and provide even stronger guarantees for the case of a cardinality constraint (Theorem 4.4.2).

Lastly, by simulating machines, we obtain in Section 4.5 a fast, practical sequential algorithm for optimizing submodular functions subject to a matroid constraint It gives a $(\frac{1}{2+e}-\epsilon)$ approximation in running time $O(\frac{n}{\epsilon}\log n)+\text{poly}(\frac{k}{\epsilon})$. (Theorem 4.5.1).

Further extending those ideas, we obtain in Chapter 5 a generic framework for distributed computation that, using a sequential algorithm, nearly recovers its approxima-

tion guarantee in a constant number of MapReduce rounds. Given an α-approximation sequential algorithm Alg, the framework guarantees an $(\alpha - O(\epsilon))$ approximation factor in $O(1/\epsilon)$ MapReduce rounds (Lemma 5.1.1 and Theorem 5.1.3). It uses the concept of *core-sets*, also present in [Mirrokni and Zadimoghaddam, 2015] for distributed submodular maximization. The techniques used therein also lead to a very simple two-round distributed algorithm for monotone maximization subject to a cardinality constraint (Theorem 5.7.1).

For the unconstrained setting, we devise in Chapter 6 parallel algorithms in the parallel transaction processing systems model. We propose algorithms that combine Random Sample and Double Greedy steps, and analyze the compromise in the approximation ratio as the degree of parallelism (i.e., transactions executed in parallel) increases.

We propose $\mathsf{Hybrid-I}$, which runs Double Greedy for t steps and samples the remaining elements, and show it obtains an approximation guarantee of $\max\left\{\frac{2+p}{8},\ \frac{p}{2}\right\}$ (Theorem 6.1.3), where $p = t/n$. We then introduce $\mathsf{Hybrid-II}$, which runs Random Sample followed by t Double Greedy steps, and prove it achieves an approximation ratio of $\left(\frac{1}{4} + \frac{p}{8} + \frac{p^2}{8}\right)$ (Theorem 6.2.1).

At a high level, the current work presents a few possible directions for future research. These include: obtaining a better comprehension of the compromise between essential components in distributed algorithms, such as approximation ratio, amount of space required or the volume of communication overhead; enhancing our algorithms in the constrained case for dealing with more general types of constraints; and finding out to what extent is randomization in fact indispensable in our results.

Generally speaking, when devising parallel algorithms, one seeks to optimize the trade-off between a few measures, such as quality of the solution obtained, amount of space used and amount of communication needed. A possible line of future work is to better understand the relationship between these components. For one, it remains open whether one can reduce the number of rounds, amount of space or amount of communication in our multi-round framework (as stated in Lemma 5.1.1). Can one devise an algorithm which attains the same $(\alpha - O(\epsilon))$ approximation factor in less than $O(1/\epsilon)$ rounds?

Likewise, it would be interesting to see new algorithms (or possibly new analyses) that improve the trade-off between parallelism and approximation guarantee obtained by the hybrid algorithms in Chapter 6 for the unconstrained case. Our analyses suggest that running Random Sample first is better, as Double Greedy could "fix" possible mistakes.

On the other hand, intuitively one could imagine that the first elements should be chosen via Double Greedy, since, by submodularity, their contribution can only decrease. It would be interesting to have this case settled. Also, no hardness results are known.

Recent works have focused on obtaining algorithms for submodular maximization under more general constraints. For instance, Chekuri et al. [2015] develop algorithms for maximizing submodular functions subject to multiple packing constraints, attaining essentially the same approximation guarantees as the Continuous Greedy algorithm. In spite of that, no distributed algorithms are known for the problem. An intriguing open question, thus, is whether our framework could be extended to handle such constraints, while maintaining the same limits on space and communication.

Lastly, randomization is a powerful tool in the design of algorithms, and plays a major role in the results obtained here. On the other hand, community has always tried to obtain derandomized counterparts of randomized algorithms, for both practical and theoretical reasons. From a theoretical perspective, it shows when is randomization inherently necessary, and limitations of the model or analysis. From a practical standpoint, it provides an often better alternative (shuffling, for instance, as done in our multi-round framework, can be computationally expensive for practical use). It is currently unclear what is the final answer on that matter for our algorithms.

On a related note, we have observed that the deterministic GreeDI algorithm of Mirzasoleiman et al. [2013] does not achieve a constant approximation in two rounds. Would it be possible to attain a constant factor in, say, three computation rounds?

Appendix A

GreeDI analysis

We give here an improved analysis of the GreeDI algorithm [Mirzasoleiman et al., 2013]. In Section A.1 we show the algorithm is in fact a $\Omega\left(1/\sqrt{k}\right)$-approximation; and in Section A.2 we show that this ratio is essentially tight.

A.1 Improved GreeDI analysis

Let OPT be an arbitrary optimal solution of k elements from V, and let M be the set of machines that have some element of OPT placed on them. For each $j \in M$ let O_j be the set of elements of OPT placed on machine j, and let $r_j = |O_j|$ (note that $\sum_{j \in M} r_j = k$). Similarly, let E_j be the set of elements returned by the Greedy algorithm on machine j. Let $e_j^i \in E_j$ denote the element chosen in the ith round of the Greedy algorithm on machine j, and let E_j^i denote the set of all elements chosen in rounds 1 through i. Finally, let $E = \cup_{j \in M} E_j$, and $E^i = \cup_j E_j^i$.

We consider the marginal values:

$$x_j^i = f_{E_j^{i-1}}(e_j^i) = f(E_j^i) - f(E_j^{i-1})$$
$$y_j^i = f_{E_j^{i-1}}(O_j),$$

for each $1 \leq i \leq k$. Note that because each element e_j^i was selected by in the ith round of the greedy algorithm on machine j, we must have

$$x_j^i \geq \max_{o \in O_j} f_{E_j^{i-1}}(o) \geq \frac{y_j^i}{r_j} \tag{A.1}$$

for all $j \in M$ and $i \in [k]$. Moreover, the sequence $x_j^1, \ldots, x_j^k$ is non-increasing for all $j \in M$. Finally, define $x_j^{k+1} = y_j^{k+1} = 0$ and $E_j^{k+1} = E_j^k$ for all j. We are now ready to prove our main claim.

Theorem A.1.1. *Let* $\tilde{\text{OPT}} \subseteq E$ *be a set of* k *elements from* E *that maximizes* f*. Then,*

$$f(\text{OPT}) \leq 2\sqrt{k} f(\tilde{\text{OPT}}).$$

Proof. For every $i \in [k]$ we have

$$\begin{aligned} f(\text{OPT}) &\leq f(\text{OPT} \cup E^i) \\ &= f(E^i) + f_{E^i}(\text{OPT}) \\ &\leq f(E^i) + \sum_{j \in M} f_{E^i}(O_j) \\ &\leq f(E^i) + \sum_{j \in M} f_{E^i_j}(O_j), \end{aligned} \tag{A.2}$$

where the first inequality follows from monotonicity of f, and the last two from submodularity of f.

Let $i \leq k$ be the smallest value such that:

$$\sum_{j \in M} r_j \cdot x_j^{i+1} \leq \sqrt{k} \cdot \left[f(E^{i+1}) - f(E^i) \right]. \tag{A.3}$$

Note that some such value must i must exist, since for $i = k$, both sides are equal to zero. We now derive a bound on each term on the right of (A.2).

Lemma A.1.2. $\sum_{j \in M} f(E^i_j) \leq \sqrt{k} \cdot f(\tilde{\text{OPT}})$.

Proof. Because i is the smallest value for which (A.3) holds, we must have

$$\sum_{j \in M} r_j \cdot x_j^{\ell} > \sqrt{k} \cdot \left[f(E^{\ell}) - f(E^{\ell-1}) \right], \text{ for all } \ell \leq i.$$

Therefore,

$$\begin{aligned}
\sum_{j\in M} r_j \cdot f(E_j^i) &= \sum_{j\in M}\sum_{\ell=1}^{i} r_j \cdot \Big[f(E_j^\ell) - f(E_j^{\ell-1})\Big] \\
&= \sum_{j\in M}\sum_{\ell=1}^{i} r_j \cdot x_j^i \\
&= \sum_{\ell=1}^{i}\sum_{j\in M} r_j \cdot x_j^i \\
&> \sum_{\ell=1}^{i} \sqrt{k} \cdot \Big[f(E^\ell) - f(E^{\ell-1})\Big] \\
&= \sqrt{k} \cdot f(E^i),
\end{aligned}$$

and so,

$$f(E^i) < \frac{1}{\sqrt{k}} \sum_{j\in M} r_j \cdot f(E_j^i) \le \frac{1}{\sqrt{k}} \sum_{j\in M} r_j \cdot f(E_j) \le \frac{1}{\sqrt{k}} \sum_{j\in M} r_j \cdot f(\tilde{\mathrm{OPT}}) = \sqrt{k} \cdot f(\tilde{\mathrm{OPT}}). \ \square$$

Lemma A.1.3. $\sum_{j\in M} f_{E_j^i(O_j)} \le \sqrt{k} \cdot f(\tilde{\mathrm{OPT}})$.

Proof. We consider two cases:

Case: $i < k$**.** We have $i+1 \le k$, and by (A.1) we have $f_{E_j^i}(O_j) = y_j^{i+1} \le r_j \cdot x_j^{i+1}$ for every machine j. Therefore:

$$\begin{aligned}
\sum_{j\in M} f_{E_j^i}(O_j) &\le \sum_{j\in M} r_j \cdot x_j^{i+1} \\
&\le \sqrt{k} \cdot (f(E^{i+1}) - f(E^i)) \\
&= \sqrt{k} \cdot f_E^i(E^{i+1} \setminus E^i) \\
&\le \sqrt{k} \cdot f(E^{i+1} \setminus E^i) \\
&\le \sqrt{k} \cdot f(\tilde{\mathrm{OPT}}).
\end{aligned}$$

Case: $i = k$**.** By submodularity of f and (A.1), we have

$$f_{E_j^i}(O_j) \le f_{E_j^{k-1}}(O_j) = y_j^k \le r_j \cdot x_j^k.$$

Moreover, since the sequence $x_j^1, \ldots, x_j^k$ is nonincreasing for all j,

$$x_j^k \leq \frac{1}{k} \sum_{i=1}^{k} x_j^i = \frac{1}{k} \cdot f(E_j).$$

Therefore,

$$\sum_{j \in M} f_{E_j^i}(O_j) \leq \sum_{j \in M} \frac{r_j}{k} \cdot f(E_j) \leq \sum_{j \in M} \frac{r_j}{k} \cdot f(\tilde{\mathrm{OPT}}) = f(\tilde{\mathrm{OPT}}).$$

Thus, in both cases, we have $\sum_{j \in M} f_{E_j^i}(O_j) \leq \sqrt{k} \cdot f(\tilde{\mathrm{OPT}})$ as required. □

Applying Lemmas A.1.2 and A.1.3 to the right of (A.2), we obtain

$$f(\mathrm{OPT}) \leq 2\sqrt{k} \cdot f(\tilde{\mathrm{OPT}}),$$

completing the proof of Theorem A.1.1. □

We have thus the following corollary. The factor $(1 - 1/e)$ comes from the (tight) approximation guarantee of Greedy under a cardinality constraint, executed in the second round.

Corollary A.1.4. *The* GreeDI *algorithm gives a* $\left(\frac{(1-1/e)}{2\sqrt{k}}\right)$*-approximation for maximizing a monotone submodular function subject to a cardinality constraint* k*, regardless of how the elements are distributed.*

A.2 A tight example for GreeDI

Here we give a family of examples that show that the GreeDI algorithm of [Mirzasoleiman et al., 2013] cannot achieve an approximation better than $1/\sqrt{k}$.

Consider the following instance of Maximum k-Coverage. We have ℓ^2+1 machines and $k = \ell + \ell^2$. Let N be a ground set with $\ell^2 + \ell^3$ elements, $N = \left\{1, 2, \ldots, \ell^2 + \ell^3\right\}$. We define a coverage function on a collection $\mathcal{S}$ of subsets of N as follows. In the following, we define how the sets of $\mathcal{S}$ are partitioned on the machines.

On machine 1, we have the following ℓ sets from OPT: $O_1 = \{1, 2, \ldots, \ell\}$, $O_2 = \{\ell + 1, \ldots, 2\ell\}$, $\ldots$, $O_\ell = \left\{\ell^2 - \ell + 1, \ldots, \ell^2\right\}$. We also pad the machine with copies of the empty set.

On machine $i > 1$, we have the following sets. There is a single set from OPT, namely $O_i' = \left\{\ell^2 + (i-1)\ell + 1, \ell^2 + (i-1)\ell + 2, \ldots, \ell^2 + i\ell\right\}$. Additionally, we

have ℓ sets that are designed to fool the greedy algorithm; the j-th such set is $O_j \cup \left\{\ell^2 + (i-1)\ell + j\right\}$. As before, we pad the machine with copies of the empty set.

The optimal solution is $O_1, \ldots, O_\ell$, $O'_1, \ldots, O'_{\ell^2}$ and it has a total coverage of $\ell^2 + \ell^3$.

On the first machine, Greedy picks the ℓ sets $O_1, \ldots, O_m$ from OPT and ℓ^2 copies of the empty set. On each machine $i > 1$, Greedy first picks the ℓ sets $A_j = O_j \cup \left\{\ell^2 + (i-1)\ell + j\right\}$, since each of them has marginal value greater than O'_i. Once Greedy has picked all of the A_j's, the marginal value of O'_i becomes zero and we may assume that Greedy always picks the empty sets instead of O'_i.

Now consider the final round of the algorithm where we run Greedy on the union of the solutions from each of the machines. In this round, regardless of the algorithm, the sets picked can only cover $\left\{1, \ldots, \ell^2\right\}$ (using the set $O_1, \ldots, O_\ell$) and one additional item per set for a total of $2\ell^2$ elements. Thus, the total coverage of the final solution is at most $2\ell^2$. Hence the approximation is at most $\frac{2\ell^2}{\ell^2+\ell^3} = \frac{2}{1+\ell} \approx \frac{1}{\sqrt{k}}$.

Bibliography

Alexander Ageev and Maxim Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8(3):307–328, 2004.

Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Proceedings of the 46th ACM Symposium on Theory of Computing (STOC)*, pages 574–583, 2014.

Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1497–1514, 2014.

Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. In *Proceedings of the 32nd ACM Symposium on Principles of Database Systems (PODS)*, pages 273–284, 2013.

Guy E. Blelloch, Richard Peng, and Kanat Tangwongsan. Linear-work greedy parallel approximate set cover and variants. In *Proceedings of the 23rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 23–32, 2011.

Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *Proceedings of the 53nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1384–1402, 2012.

Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1433–1452, 2014a.

Niv Buchbinder, Moran Feldman, and Roy Schwartz. Comparing apples and oranges: Query tradeoff in submodular maximization. In *Proceedings of the 26th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1149–1168, 2014b.

Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint. In *Proceedings of the 12th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 182–196, 2007.

Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint. *SIAM Journal of Computing*, 40(6):1740–1766, 2011.

Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *Proceedings of the 51st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 575–584, 2010.

Chandra Chekuri, T. S. Jayram, and Jan Vondrák. On multiplicative weight updates for concave and submodular function maximization. In *Proceedings of the 6th Innovations in Theoretical Computer Science (ITCS)*, pages 201–210, 2015.

Flavio Chierichetti, Ravi Kumar, and Andrew Tomkins. Max-cover in map-reduce. In *Proceedings of the 19th International World Wide Web Conference (WWW)*, pages 231–240, 2010.

Rafael da Ponte Barbosa, Alina Ene, Huy Le Nguyen, and Justin Ward. The power of randomization: Distributed submodular maximization on massive datasets. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1236–1244, 2015.

Rafael da Ponte Barbosa, Alina Ene, Huy Le Nguyen, and Justin Ward. A new framework for distributed submodular maximization. In *Proceedings of the 57th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 645–654, 2016.

Shahar Dobzinski and Jan Vondrák. From query complexity to computational complexity. In *Proceedings of the 44th ACM Symposium on Theory of Computing (STOC)*, pages 1107–1116, 2012.

Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45: 634–652, 1998.

Uriel Feige, Vahab S Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal of Computing*, 40(4):1133–1153, 2011.

Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 570–579, 2011.

Yuval Filmus and Justin Ward. A tight combinatorial algorithm for submodular maximization subject to a matroid constraint. In *Proceedings of the 53nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 659–668, 2012.

Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the mapreduce framework. In *Algorithms and Computation*, pages 374–383. Springer, 2011.

Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *Proceedings of the 6th Conference Web and Internet Economics (WINE)*, pages 246–257, 2010.

Dirk Hausmann and Bernhard Korte. k-greedy algorithms for independence systems. *Mathematical Methods of Operations Research*, 22:219–228, 1978.

Thomas A. Jenkyns. The efficacy of the "greedy" algorithm. In *Proceedings of the 7th Southeastern Conference on Combinatorics, Graph Theory, and Computing*, pages 341–350, 1976.

Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for mapreduce. In *Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 938–948, 2010.

Andreas Krause and Carlos Guestrin. Near-optimal observation selection using submodular functions. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI)*, pages 1650–1654, 2007.

Ravi Kumar, Benjamin Moseley, Sergei Vassilvitskii, and Andrea Vattani. Fast greedy algorithms in mapreduce and streaming. In *Proceedings of the 25th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 1–10, 2013.

Hsiang-Tsung Kung and John T. Robinson. On optimistic methods for concurrency control. *ACM Transactions on Database Systems (TODS)*, 6(2):213–226, 1981.

László Lovász. Submodular functions and convexity. In *Mathematical programming: the state of the art*, pages 235–257. Springer, 1983.

Vahab S. Mirrokni and Morteza Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. In *Proceedings of the 47th ACM Symposium on Theory of Computing (STOC)*, pages 153–162, 2015.

Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Proceedings of the 27th Conference on Neural Information Processing Systems (NIPS)*, pages 2049–2057, 2013.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1812–1818, 2015.

George L. Nemhauser and Laurence A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of Operations Research*, 3(3): 177–188, 1978.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978a.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions—ii. *Mathematical Programming Studies*, 8:73–87, 1978b.

M. Tamer Özsu and Patrick Valduriez. *Principles of distributed database systems*. Springer Science & Business Media, 2011.

Xinghao Pan, Stefanie Jegelka, Joseph E Gonzalez, Joseph K Bradley, and Michael I Jordan. Parallel double greedy submodular maximization. In *Proceedings of the 28th Conference on Neural Information Processing Systems (NIPS)*, pages 118–126, 2014.

Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2003.

Athanasios Tsanas, Max A Little, Patrick E McSharry, and Lorraine O Ramig. Enhanced classical dysphonia measures and sparse regression for telemonitoring of parkinson's disease progression. In *Proceedings of the 35th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 594–597, 2010.

Jan Vondrák. Symmetry and approximability of submodular maximization problems. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 651–670, 2009.

Laurence A. Wolsey. Maximising real-valued submodular functions: Primal and dual heuristics for location problems. *Mathematics of Operations Research*, 7(3):pp. 410–425, 1982.